დალი მაგრაქველიძე

ღრუბლოვანი და გრიდ კომპიუტინგი

სალექციო კურსი

2019 წ

სალექციო კურსის მიზანია: შეასწავლოს სტუდენტებს ღრუბლოვანი და გრიდ გამოთვლების არსი და მისი ტექნოლოგია; გააცნოს გამოთვლითი ტექნიკის განვითარების ძირითადი ეტაპების; გააანალიზოს აპარატული უზრუნველყოფის განვითარების თანამედროვე ტენდენციები, რომლებსაც მივყავართ ღრუბლოვანი გამოთვლების ტექნოლოგიების წარმოშობამდე.

ღრუბლოვანი გამოთვლები წარმოადგენენ ხელმისაწვდომობის დინამიკურ, მასშტაბურ საშუალებას გარეშე გამომთვლელ რესურსებთან სერვისის სახით, რომელიც წარმოდგენილია ინტერნეტის საშუალებით, ამასთან მომხმარებელს არ მოეთხოვება „ღრუბლის“ ინფრასტუქტურის შესახებ რაიმე განსაკუთრებული ცოდნა ან ამ „ღრუბლოვანი“ ტექნოლოგიის მართვის უნარი.

დასახული მიზნის მიღწევისათვის წინამდებარე სალექციო კურსის შესწავლის ძირითადი საგანია ღრუბლოვანი ვირტუალიზაცია და ვებ-სერვისების მიმოხილვა, რომელიც წარმოდგენილია ღრუბლოვანი გამოთვლების კონცეფციით.

ყოველ თეორიულ მსჯელობას თან ახლავს პრაქტიკული მაგალითები, რომლებიც საშუალებას იძლევიან ღრუბლოვანი ვირტუალიზაციის არსის სრულად გაგებას.

სალექციო კურსი სასარგებლო იქნება სტუდენტებისა და ინფორმატიკის პრობლემებით დაინტერესებული ყველა პირისათვის.

რედაქტორი:

ვახტანგ კვარაცხელია - საქართველოს ტექნიკური უნივერსიტეტი, პროფესორი.

რეცენზენტები:

მაქსიმ იავიჩი - კავკასიის უნივერსიტეტის, ტექნოლოგიების სკოლის პროფესორი, კიბერ უსაფრთხოების მიმართულების ხელმძღვანელი.

ავთანდილ გაგნიძე - შავი ზღვის საერთაშორისო უნივერსიტეტის რექტორის მრჩეველი, პროფესორი.

ISBN 978-9941-8-1842-4

თემა 1. შესავალი 4
გრიდი, ღრუბელი და ვირტუალიზაცია 4
თემა 2. განაწილებული გამოთვლების ევოლუცია 12
განაწილებული სისტემების თვისებები 23
თემა 3. *ინტერნეტის საშუალებით მასშტაბური გამოთვლები – ქსელური სისტემებისათვის ტექნოლოგიები – თანამედროვე კომპიუტერებისათვის კლასტერები. ბლეიდ სერვერები* 31
თემა 4.მონაცემთა შენახვის ქსელები 48
SAN კომუტატორების არსი 49
ვირტუალური SAN 49
ერთიანი SAN 50
კონვერგენტული SAN 50
SAN-ის პლიუსები და მინუსები 50
SAN NAS-ის წინააღმდეგ 51
მთავარი გამყიდველები და პროდუქტები 52
SAN საცავის უპირატესობა 53
ფაილი ბლოკის წინააღმდეგ 55
iSCSI SAN 56
SAN ტოპოლოგიები 59
ერთი-სტრუქტურა ერთი-კვანძი კონფიგურაცია 60
აპლიკაციის კონსოლიდაცია 62
თემა 5. ვირტუალიზაციის ტექნოლოგია ვირტუალიზაცია ღრუბლოვან გამოთვლებში 64
როგორ მუშაობს ვირტუალიზაცია ღრუბლოვან გამოთვლებში? 65
ვირტუალზიაციის ტექნოლოგია 66
ვირტუალიზაციის უპირატესობა 70
ვირტუალიზაციის ნაკლოვანებები 72
თემა 6. სერვერების ვირტუალიზაცია 77
თემა 7. ღრუბლოვანი გამოთვლების საფუძვლები 80
ღრუბლოვანი გამოთვლის სახეები 81
ღრუბლოვანი გამოთვლების ღირსებები 92
ღრუბლოვანი გამოთვლების ნაკლი და პრობლემები 95
ღრუბლოვანი სერვისების სახეები: IaaS, PaaS, SaaS, FaaS 99
ღრუბლოვანი გამოთვლების გამოყენება 100
რა არის ღრუბლოვანი გამოთვლები 101
ღრუბლოვანი გამოთვლების ჩაშლა(BREAKING DOWN) 101
თემა 8.ღრუბელში ვებ-სერვისები 108
ვებ სერვისები და ღრუბლოვანი გამოთვლები 108
ინფრასტუქტურა როგორც სერვისი (IaaS) 108
პლატფორმა როგორც სერვისი (PaaS) 116
პროგრამული უზრუნველყოფა როგორც სერვისი (SaaS) 126
კომუნიკაცია როგორც სერვისი (CaaS) 131
მონოტორინგი როგორც სერვისი (MaaS) 134
ღრუბლოვანი გამოთვლების ძირითადი მახასიათებლები 138

თემა 9. Windows Azure SDK 141
თემა 10. Azure Service Platform 149
არქიტექტურა Windows Azure Platform 149
Windows Azure Storage 152
Azure Table Services 153
თემა 11. უსაფრთხოება ღრუბელში 163
სიძნელეები, რომლებიც დაკავშირებულია საკანონმდებლო, ორგანიზაციულ-სამართლებრივ ასპექტებთან და სტანდარტიზაციის საკითხებთან 164
მონაცემთა უსაფრთხოება 165
დაშიფრეთ ყველაფერი! 170
ქსელური ტრაფიკის დაშიფვრა 170
ფაილური სისტემების დაშიფვრა 171
სტანდარტების და ნორმატიულ -საკანონმდებლო აქტებთან შესაბამისობა 173
ქსელური უსაფრთხოება 177
ქსელური თავდასხმების აღმომჩენი სისტემები 184
ჰოსტების დაცვა 188
სისტემის დაცვის გაძლიერება 189
მონაცემთა სეგმენტაცია 193
გამოყენებული ლიტერატურა: 198

თემა 1. შესავალი

გრიდი, ღრუბელი და ვირტუალიზაცია

ღრუბლოვანი გამოთვლების ქვეშ ჩვეულებრივ იგულისხმება ქსელიდან საჭირო გამოთვლითი სიმძლავრეების მიღების შესაძლებლობა, ამასთან მომხმარებელისთვის მთავარი არაა ამ მექანიზმის რეალიზაციის დეტალები და ის იღებს ამ „ღრუბლიდან“ ყველაფერს საჭიროს. ამის ნათელ მაგალითად შეიძლება გამოდგეს საძიებელი სისტემების ინტერფეისი, რომლებიც ძალიან მარტივია, მაგრამ ამავდროულად მომხმარებელს საჭირო ინფორმაციის მოსაძებნად უზარმაზარ გამოთვლით რესურსებს აძლევენ. ამჟამად მსხვილი გამოთვლითი ცენტრები არა მხოლოდ იმის საშუალებას იძლევიან, რომ შეინახონ და გადაამუშავონ თავის შიგნით განსაზღვრული მონაცემები, არამედ ასევე საკუთარი ვირტუალური დატა-ცენტრების შექმნის საშუალებას იძლევიან, რომელიც ახალგაზრდა კომპანიებს ეხმარება ზედმეტი ძალისხმევის გარეშე ნულიდან შექმნან თავიანთი ინფრასტუქტურა. ამჟამად, არსებობს „ღრუბლოვანი გამოთვლების“ განსაზღვრული სიმრავლე. უფრო ხშირად ისინი განსხვავდებიან თავიანთი მნიშვნელობებით და აქცენტებით. განვიხილოთ ამ განსაზღვრებებიდან ზოგიერთი იმისათვის, რომ გავერკვეთ რას ნიშნავს „ღრუბლოვანი“ გამოთვლები სხვადასხვა თავალსაზრისით.

განსხვავება გრიდ გამოთვლებსა და ღრუბლოვან გამოთვლებს შორის

გრიდ გამოთვლები და ღრუბლოვანი გამოთვლები კონცეპტუალრად მსგავსია და ადვილად შეიძლება აგვერიოს ერთმანეთში. მათი არსი თითქმის ერთნაირია, ორივე იყენებს მომხმარებლებისთვის მომსახურების გაწევის ერთიდაიგივე მეთოდს - რესურსების განაწილებას მომხმარებელბის დიდ რაოდენობას შორის. ორივე დაფუძნებულია ქსელურ ტექნოლოგიაზე და შეუძლია მრავალი ამოცანის შესრულება, რაც ნიშნავს იმას, რომ მომხმარებელს შეუძლია მიმართოს ერთ ან რამოდენიმე აპლიკაციას სხვადასხვა ამოცანის შესასრულებლად.

გრიდ გამოთვლები მოიცავს გამომთვლელი რესურსების ვირტუალიზაციას დიდი მოცულობის მონაცემების შესანახად, ღრუბლოვან გამოთვლების დროს აპლიკაცია არ

მიმართავს რესურსებს პირდაპირ, არამედ მიმართვა ხდება მომსახურების მეშვეობით ინტერნეტის გავლით.

გრიდ გამოთვლებში რესურსები ნაწილდება ბადეებში, მაშინ როდესაც ღრუბლოვანი გამოთვლების შეთხვევაში რესურსები იმართება ცენტრალიზებულად. მოდით მოკლედ გადავავლოთ თვალი გამოთვლის ტექნოლოგიის ორ მეთოდს.

რა არის გრიდ გამოთვლები?

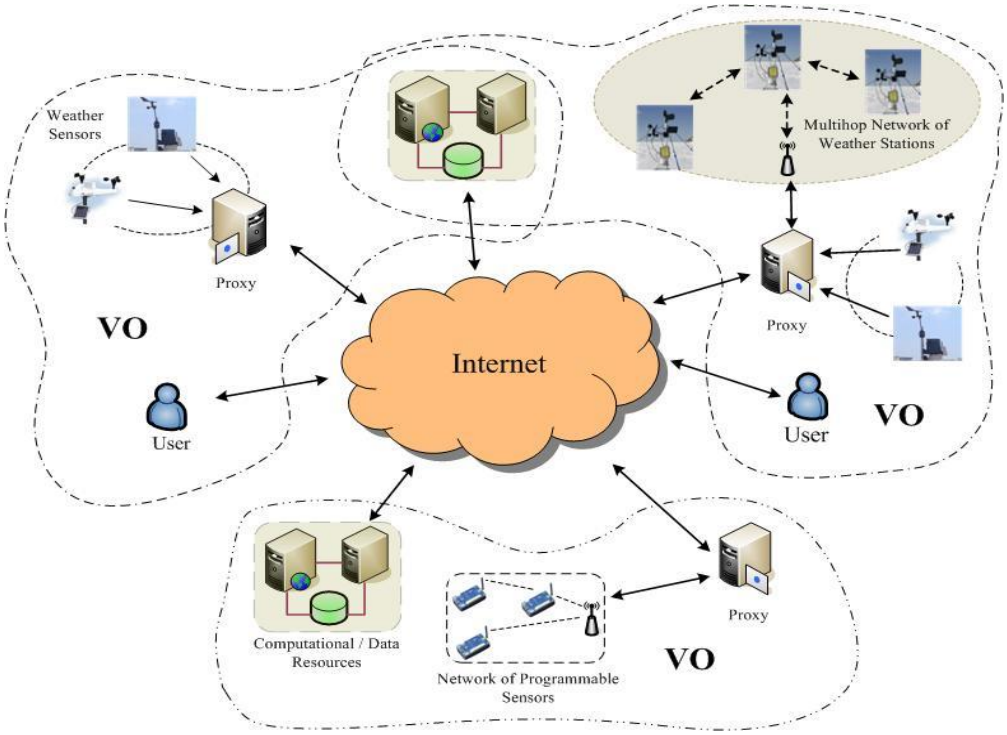
გრიდ გამოთვლები ესაა ქსელზე დაფუძნებული გამოთვლითი მოდელი, რომელსაც აქვს დიდი მოცულობის მონაცემებზე მუშაობის შესაძლებლობა რათა დაეხმაროს ქსელური კომპიუტერების ჯგუფს მოახდინონ კოორდინირება პრობლემის ერთად გადასაჭრელად.

ძირითადად, ეს არის საერთო პრობლემაზე მომუშვე ერთმანეთთან კავშირში მყოფი კომპიუტერების დიდი ქსელი, რომელიც რამდენიმე მცირე ზომის ერთეულადაა დაყოფილი, რომელსაც გრიდი ეწოდება. ის ეფუძნება განაწილებულ არქიტექტურას, რაც იმას ნიშნავს, რომ ამოცანები იმართება და დაგეგმილია გადანაწილების გზით დროზე დამოკიდებულების გარეშე.

კომპიუტერების ჯგუფი მოქმედებს, როგორც ვირტუალური სუპერკომპიუტერი რომელიც უზრუნველყოფს სკალირებულ და შეუფერხებელ წვდომას გლობალურ გამომთვლელ რესურსებთან, რომლებიც გეოგრაფიულად განაწილებულია და წარმოადგენენ მას, როგორც ერთიან უნიფიცირებულ რესურსს მსხვილმასშტაბიანი აპლიკაციების შესასრულებლად, დიდი მომაცემების ანალიზი.

რა არის ღრუბლოვანი გამოთვლები?

ღრუბლოვანი გამოთვლები ეს არის ინტერნეტზე დაფუძნებული ტიპის გამოთვლები, სადაც აპლიკაცია არ მიმართავს რესურსებს პირდაპირ, თუმცა იგი გაზიარებული რესურსებისაგან ქმნის უდიდეს რესურსების ბაზას. ეს არის თანამედროვე გამოთვლითი პადაგრიმა დაფუძნებული ქსელურ ტექნოლოგიაზე, რომელიც სპეციალურად შექმნილია სკალირებული და გაზომილი IT რესურსების დაშორებულად უზრუნველსაყოფად.



ნახ.1. ღრუბლის სქემატური გამოსახვა.

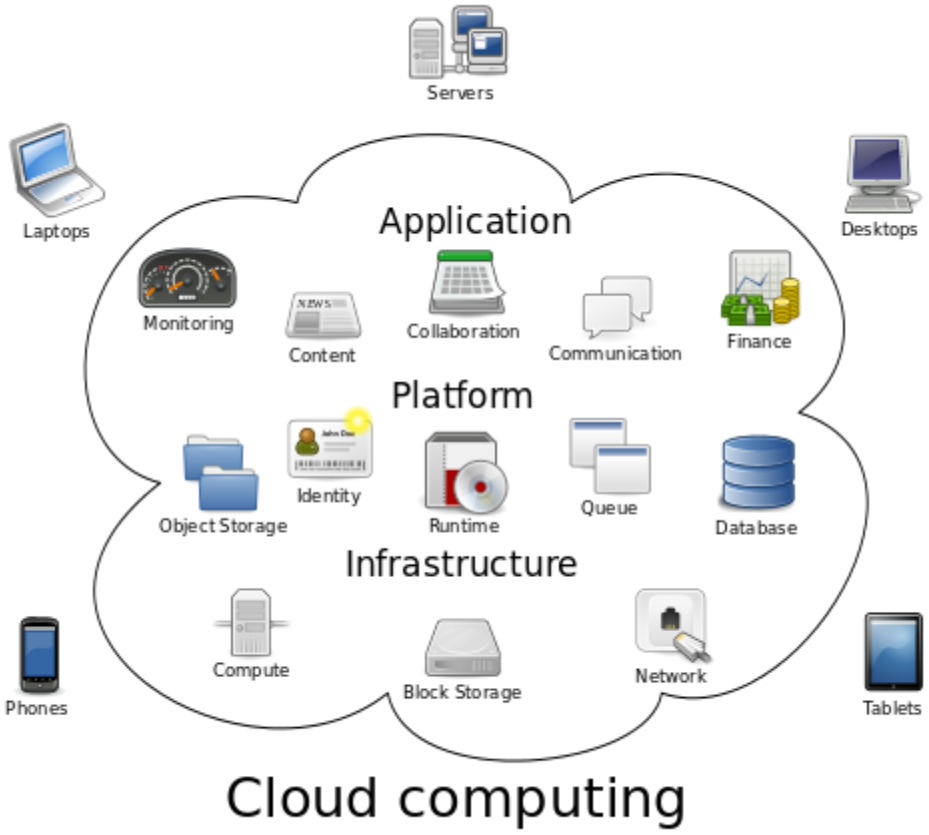
ის უზრუნველყოფს მოთხოვნით წვდომას დინამიურად კონფიგურირებული გამოთვლითი რესურსების გაზიარებულ ბაზასთან და ამით მაღალი დონის მომსახურება აღმოფხვრის ადგილობრივ ინფრასტრუქტურაში დიდი ოდენობით ინვესტიციების საჭიროებას. კომპიუტერული რესურსები იმართება ცენტრალიზებულად, რომელიც განთავსებულია კლასტერში მრავალ სერვერზე. მომხმარებლებს აქვთ საშუალება მიმართონ პროგრამებს და აპლიკაციებს იქიდან, საიდანაც მათთვის მოსახერხებელია საკუთარი მონაცემების შენახვაზე ზრუნვის გარეშე.

განსხვავება გრიდ კომპიუტინგსა და ღრუბლოვან კომპიუტინგს შორის

1. გრიდ კომპიუტინგსა და ღრუბლოვან კომპიუტინგში გამოყენებული ტექნოლოგია

გრიდ კომპიუტინგი არის გამოთვლის ისეთი ფორმა, რომელიც გამომდინარეობს დანაწილებული აქიტექტურიდან. რაც ნიშნავს, რომ ცალკეული ერთეული ამოცანა დაყოფილია რამოდენიმე მცირე ამოცანად დანაწილებული სისტემის გავლით, რომელიც მოიცავს მრავალი კომპიუტერისაგან შემდგარ ქსელს.

მეორე მხრივ, ღრუბლოვანი კომპიუტინგი არის გამოთვლის მთლიანად ახალი კლასი დაფუძნებული ქსელურ ტექნოლოგიაზე, სადაც ღრუბლის ყოველ მომხმარებელს აქვს თავისი საკუთარი რესურსები, რომელსაც აწვდის სპეციალური მომსახურების პროვაიდერი.



ნახ.2. ღრუბლოვანი გამოთვლები.

2. გრიდ კომპიუტინგისა და ღრუბლოვანი კომპიუტინგის ტერმინოლოგია

ორივე ქსელზე დაფუძნებული გამოთვლით ტექნოლოგიაა, რომელებიც იზიარებენ ერთნაირ მახასიათებლებს, როგორიცაა რესურსების გაერთიანება, თუმცა ისინი ძალიან განსხვავდებიან ერთმანეთისაგან არქიტექტურის, ბიზნეს მოდელის, თავსებადობისა და სხვა თვალსაზირსით. გრიდ კომპიუტინგი არის ერთი ამოცანის გადაჭრაზე მომუშავე კომპიუტერული რესურსების ერთობლიობა სხვადასხვა ადგილმდებარეობით. გრიდი მოქმედებს როგორც განაწილებული სისტემა რესურსების ერთობლივი გაზიარებისათვის. მეორე მხრივ, ღრუბლოვანი კომპიუტინგი არის გამოთვლის ფორმა დაფუძნებული ვირტუალიზებულ რესურსებზე, რომელიც განთავსებულია მრავალ ადგილზე კლასტერებში.

3. კომპიუტერული რესურსები გრიდ კომპიუტინგში და ღრუბლოვან კომპიუტინგში

გრიდ კომპიუტინგი ეფუძნება განაწილებულ სისტემას, რაც ნიშნავს, რომ გამოთვლითი რესურსების განაწილებულია სხვადასხვა კომპიუტერულ ერთეულებს შორის, რომელიც განთავსებულია სხვადასხვა საიტებს, ქვეყნებსა და კონტინენტებს შორის. ღრუბლოვან

კომპიუტიუნგში გამოთვლითი რესურსები, რომელიც განთავსებულია ღრუბლის პროვაიდერების პირად მონაცემთა ცენტრში მრავალი სერვერისგან შემდგარ კლასტერში, იმართება ცენტრალიზებულად.

4. კვლევითი საზოგადოება

გრიდ გამოთვლებში კომპიუტერული რესურსები წარმოდგენილია როგორც კომპიუტერული პლატფორმა, რომელიც განაწილებულია გეოგრაფიულად და დაჯგუფებულია ვირტუალურ ორგანიზაციაში მრავალი მომხმარებლის საზოგადოებით ფართომასშტაბიანი პრობლემის ინტერნეტის მეშვეობით გასადაჭრელად. გრიდი მოიცავს უფრო მეტ რესურს ვიდრე უბრალოდ კომპიუტერები და ქსელები. მეორე მხრივ, ღრუბლოვანი კომპიუტინგი მოიცავს სისტემურ ადმინისტრატორთა ერთობლივ ჯგუფს, რომლებიც მართავენ მთელ დომენს

5. გრიდ კომპიუტინგისა და ღრუბლოვანი კომპიუტენგის ფუნქციები

გრიდ კომპიუტინგის მთავარი ფუნქციაა საქმის განრიგის გაკეთება ყველა სახის გამოთვლითი რესურსის გამოყენებით, სადაც ამოცანა დაყოფილია რამოდენიმე დამოუკიდებელ ქვეამოცანად და თითოეული მანქანა გრიდში შეესაბამება ამოცანას. მას შემდეგ რაც ყველა ქვეამოცანა შესრულდება ისინი უკან უბრუნდება მთავარ მანქანას, რომელიც ამუშავებს ყველა ამოცანას. ღრუბლოვანი კომპიუტინგი იყენებს სერვერების კლასტერებიდან რესურსების შეგროვებას რესურსების დაჯგუფების მეშვეობით, საჭიროებაზე დაყრდნობით.

6. გრიდ კომპიუტინგისა და ღრუბლოვანი კომპიუტინგის აპლიკაციები

ღრუბლოვან კომპიუტინგში ტერმინი „ღრუბელი“ აღნიშნავს ინტერნერს და მთლიანობაში იგი ნიშნავს ინტერნეტზე დაფუძნებულ გამოთვლებს. ღრუბელი მართავს მონაცემებს, უსაფრთოხების მონაცემებს, სამუშაოს რიგითობას და სხვას მოწყობილობისა და პროგრამების შეძენას, რომელიც საჭიროა აპლიკაციის შესაქმნელად, საჭიროებისა და სირთულის აღმოფხვრის გზით.

დღესდღესობით „ღრუბლოვანი“ გამოთვლების ტექნოლოგია თანდათან დიდ პოპულარობას იძენს, ხოლო $Cloud\ Computing$ კონცეფცია წარმოადგენს ინფორმაციული ტექნოლოგიების განვითარებაში ერთ-ერთ ყველაზე მოდურ ტენდენციას. $Gartner$ -ის შეფასებით „ღრუბელი“ – ბიზნესის ერთ-ერთ მთავარ პრიორიტეტს წარმოადგენს. მსოფლიო

IT ვენდორები (*Microsoft, Amazon, Google* და სხვები) ამა თუ იმ ფორმით ნერგავენ „ღრუბლოვანი“ გამოთვლების სერვისებს.

საბოლოოდ, ღრუბლოვანი გამოთვლები წარმოადგენენ ხელმისაწვდომობის დინამიურ, მასშტაბურ საშუალებას გარეშე გამომთვლელ რესურსებთან სერვისის სახით, რომელიც წარმოდგენილია ინტერნეტის საშუალებით, ამასთან მომხმარებელს არ მოეთხოვება „ღრუბლის“ ინფრასტუქტურის შესახებ რაიმე განსაკუთრებული ცოდნა ან ამ „ღრუბლოვანი“ ტექნოლოგიის მართვის უნარი.

Cloud Computing* - ეს არის პროგრამულ-აპარატურული უზრუნველყოფა, რომელიც მომხმარებელს ინტერნეტის ან სერვისის სახით ლოკალური ქსელის მეშვეობით საშუალებას აძლევს გამოიყენოს მოსახერხებელი ინტერფეისი გამოყოფილ სერვისებზე (გამომთვლელ რესურსებზე, პროგრამებზე და მონაცემებზე) დისტანციური ხელმისაწვდომობისათვის.** მომხმარებლის კომპიუტერი ამ შემთხვევაში გამოდის ქსელში მიერთებული რიგითი ტერმინალის როლში. კომპიუტერებს, რომლებიც ახორციელებენ ***Cloud Computing -ს უწოდებენ „გამომთვლელ ღრუბელს“. ამასთან „გამომთვლელ ღრუბელში“ შემავალ კომპიუტერებს შორის დატვირთვა ავტომატურად ნაწილდება.

ღრუბლოვანი გამოთვლები – ეს არის ახალი მიდგომა, რომელიც ფართო სპექტრის ეფექტური ტექნოლოგიების გამოყენების მეშვეობით იმ *IT* სისტემების სითულის დაწევის საშუალებას იძლევა, რომლებიც იმართება დამოუკიდებლად და ხელმისაწვდომია ვირტუალური ინფრასტუქტურის ჩარჩოებში მოთხოვნების მიხედვით, ასევე გამოყენებულია სერვისის ხარისხში. კერძო ღრუბლებზე გადასვლისას შემკვეთებს შეუძლიათ მიიღონ რიგი უპირატესობა, რომელშიც შედის: ***IT*** -ზე ფასდაკლება, სერვისის წარმოდგენის ხარისხის და ბიზნესის დინამიურობის ამაღლება.

„ღრუბელი“ წარმოადგენს ინფორმაციული მომსახურებების წარდგენისა და მიღების ახალ ბიზნეს-მოდელს. ეს მოდელი გვპირდება ოპერატიული და კაპიტალური ხარჯების დაწევას. ის IT დეპარტამენტებს საშუალებას აძლევს კონცენტრირება მოახდინონ სტრატეგიულ პროექტებზე და არა მონაცემთა დამუშავების საკუთარი ცენტრების მართვის რუტინულ ამოცანებზე.

ღრუბლოვანი გამოთვლები – ეს არამარტო *IT* -ში ტექნოლოგიური ინოვაციებია, არამედ ახალი ბიზნეს-მოდელების შექმნის საშულებაცაა, როცა *IT* -პროდუქტის მცირე მწარმოებლებს,

მათ შორის რეგიონებშიც, ბაზარზე თავისი მომსახურების სწრაფი შეთავაზების და თავისი ბიზნეს–იდეის მცირე დანახარჯებით ხორცშესხმის შესაძლებლობა უჩნდებათ. ღრუბლოვანი გამოთვლების მხარდაჭერა დამწყებ კომპანიებში, ინვესტიციებთან შეხამებით, ქმნის სწრაფად განვითარებად ინოვაციური წარმოების ეკოსისტემას.

ღრუბლოვანი გამოთვლები წამოადგენს საბაზრო პასუხს სისტემატურ სპეციალიზაციასა და *IT* –ში აუთსორსინგის როლის გაზრდაზე. თავისი არსით, ღრუბლოვან გამოთვლებზე გადასვლა ნიშნავს *IT*-ს მართვის ტრადიციული პროცესების აუთსორსინგს, პროფესიონალი, გარე მომწოდებლების ინფრასტრუქტურით. ღრუბლოვანი გამოთვლების სფეროში გადაწყვეტილებათა თანამედროვე მიმწოდებელთა უმეტესობა არა მარტო არსებული ღრუბლოვან პლატფორმების გამოყენების, არამედ საკუთარის შექმნის საშუალებას იძლევიან, რომლებიც შემკვეთების ტექნოლოგიურ და იურიდიულ მოთხოვნებს პასუხობენ.

„ღრუბლოვანი გამოთვლები“ შემდეგნაირად მუშაობს: აპლიკაციის გასაშვებად საკუთარი სერვერების ყიდვის, დაყენების და მართვის მაგივრად ხდება $Microsoft, Amazon, Google$ და სხვა კომპანიებისაგან სერვერების იჯარით აღება. შემდეგ მომხმარებელი მართავს თავის იჯარით აღებულ სერვერებს ინტერნეტის მეშვეობით, იხდის რა ამ დროს მხოლოდ მათი მონაცემების გადამუშავების და მონაცემთა შენახვა-გამოყენებისათვის საფასურს. გამომთვლელი ღრუბელი დატაცენტრებში განლაგებული ათასი სერვერისაგან შედგება, რომლებიც უზრუნველყოფენ ათობით ათასი აპლიკაციის მუშაობას, რომლებსაც ერთდროულად მილიონობით მოხმარებელი იყენებს. ასეთი მსხვილმასშტაბიანი ინფრასტრუქტურის ეფექტური მართვის აუცილებელ პირობას წარმოადგენს მაქსიმალურად სრული *ავტომატიზაცია*. გარდა ამისა, სხვადასხვა მომხმარებლის უზრუნველსაყოფად – ღრუბლოვანი ოპერატორების, სერვერ-პროვაიდერების, შუამავლების, *IT* - ადმინისტრატორების, აპლიკაციის მომხმრებლების – გამოთვლით რესურსებზე დაცული ხელმისაწვდომობისათვის ღრუბლოვანი ინფრასტრუქტურა უნდა ითვლისწინებდეს თვითმართველობის შესძლებლობას და უფლებამოსილებილებების დელიგირებას.

„ღრუბლოვანი“ გამოთვლების კონცეფცია ცარიელ ადგილას არ წარმოიშობილა, არამედ წარმოადენს ბოლო რამდენიმე ათეული წლის მანძილზე ინფორმაციული ტექნოლოგიური ევოლუციის შედეგს და თანამედროვე ბიზნესის გმოწვევებზე პასუხს. $Gartner\ Group$ -ის

ანალიტიკოსები „ღრუბლოვან“ გამოთვლებს უწოდებენ მომავლის ყველაზე პერსპექტიულ სტრატეგიულ ტექნოლოგიებს, აკეთებენ რა ტექნოლოგიების ინფორმაციის დიდი ნაწილის „ღრუბელში“ გადასვლის პროგნოზირებას 5-7 წლის განმავლობაში.

თემა 2. განაწილებული გამოთვლების ევოლუცია

განაწილებული გამოთვლები - ეს არის კომპიუტერული მეცნიერების ის სფერო, რომელიც შეისწავლის სისტემების განაწილებას. **განაწილებული სისტემა** - ეს არის სისტემა, რომლის კომპონენტები განლაგებულია სხვადასხვა ქსელურ კომპიუტერებზე, რომლებიც ახდენენ თავიანთი მოქმედებების ურთიერთქმედებას და კოორდინირებას ერთმანეთთან შეტყობინებების გადაცემით. კომპონენტები ურთიერთქმედებენ ერთმანეთთან საერთო მიზნის მისაღწევად.

განაწილებული სისტემების სამი მნიშვნელოვანი მახასიათებელია: კომპონენტების შერწყმა, გლობალური საათების ნაკლებობა და კომპონენტების დამოუკიდებელი გაუმართაობა.

განაწილებული სისტემის მაგალითები ვარირებენ SOA სისტემაზე დაფუძნებული სისტემებიდან მასობრივ მრავალმოთამაშიან ონლაინ-თამაშებამდე და ერთრანგიან აპლიკაციებამდე. კომპიუტერულ პროგრამას, რომელიც მუშაობს განაწილებულ სისტემაში, **განაწილებული პროგრამა** ეწოდება (ხოლო განაწილებული პროგრამირება - ეს არის ასეთი პროგრამების დაწერის პროცესი).

შეტყობინების გადაცემის მექანიზმისათვის არსებობს რეალიზაციის მრავალი ტიპი HTTP, RPC (დაშორებული პროცედურების გამოძახება)-ის მსგავსი კონექტორების და გზავნილთა რიგების ჩათვლით.

განაწილებული გამოთვლები ასევე მიეკუთვნება კომპიუტერული პრობლემების გადაჭრის მიზნით განაწილებული სისტემების გამოყენებას. განაწილებულ გამოთვლებში, პრობლემა დაყოფილია ბევრ ამოცანად, რომელთაგან თითოეული გადაწყდება ერთი ან მეტი კომპიუტერით, რომელიც ერთმანეთთან ურთიერთობენ გზავნილის მეშვეობით.

განაწილებული სისტემები - ეს არის ქსელური კომპიუტერების ჯგუფები, რომელთაც მუშაობის ერთი და იგივე მიზანი აქვთ. ტერმინები "თანხვედრი გამოთვლები", "პარალელური გამოთვლები" და "განაწილებული გამოთვლები" ბევრ რამეში ემთხვევა ერთმანეთს და მათ შორის მკაფიო განსხვავება არ არსებობს.

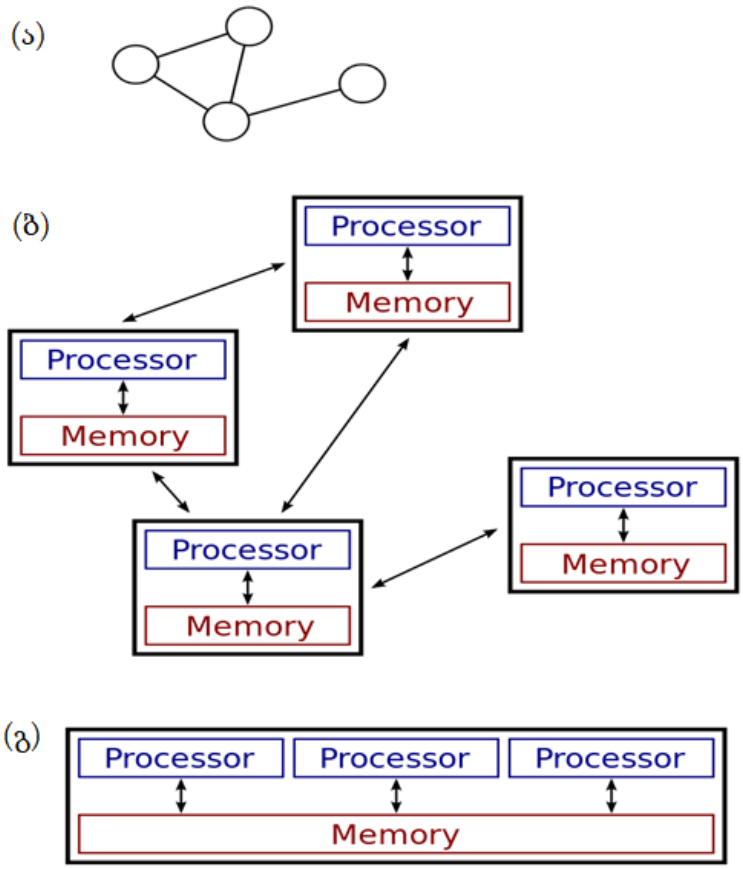
იგივე სისტემა შეიძლება დავახასიათოთ, როგორც "პარალელური" და "განაწილებული"; ტიპიური განაწილებული სისტემის პროცესორები მუშაობას თანხვედრილად, პარალელურად აწარმოებენ. პარალელური გამოთვლით შეიძლება განიხილებოდეს როგორც განაწილებული გამოთვლები, და განაწილებული გამოთვლები შეიძლება განიხილებოდეს, როგორც პარალელური გამოთვლების სუსტად დაკავშირებული კომპიუტერული ფორმები. მიუხედავად ამისა, შესაძლებელია შემდეგი კრიტერიუმების გამოყენებით თანხვედრი სისტემების, "პარალელურად" ან "განაწილებულად"კლასიფიცირება:

• პარალელურ გამოთვლებში, ყველა პროცესორს ერთმანეთს შორის ინფორმაციის გაცვლისათვის შეუძლია ჰქონდეს წვდომა საერთო მეხსიერებაზე.

• განაწილებულ გამოთვლებში, თითოეული პროცესორის აქვს საკუთარი პირადი მეხსიერების ნაწილი (დისტრიბუტირებული მეხსიერება). ინფორმაცია იცვლება პროცესორებს შორის შეტყობინებების გავლით.

ნახატ 3-ზე მარჯვნივ გამოსახულია განაწილებულ და პარალელურ სისტემებს შორის განსხვავება. ნახ. (ა) არის ტიპური განაწილებული სისტემის სქემატური ხედვა; სისტემა წარმოადგენს ქსელის ტოპოლოგიას, რომელშიც თითოეული კვანძი წარმოადგენს კომპიუტერს, ხოლო კვანძების დამაკავშირებელი თითოეული ხაზი - კომუნიკაციის ბმულია. ნახ. (ბ)-ზე ნაჩვენებია იგივე განაწილებული სისტემა უფრო დეტალურად: თითოეულ კომპიუტერს აქვს საკუთარი ლოკალური მეხსიერება და ინფორმაციის გაცვლა შესაძლებელია განხორციელდეს მხოლოდ ერთი კვანძისგან მეორეზე გადასასვლელად შეტყობინებების არსებული კავშირების გამოყენებით. ნახ. (გ) გვიჩვენებს პარალელურ სისტემას, რომელშიც თითოეულ პროცესორს აქვს პირდაპირი წვდომა საერთო მეხსიერებასთან.

ვითარება კიდევ უფრო რთულდება პარალელური და განაწილებული *ალგორითმის* ტერმინების ტრადიციული გამოყენების შედეგად, რომელიც არ შეესაბამება პარალელური და განაწილებული სისტემების ზემოაღნიშნულ განსაზღვრებებს (უფრო დეტალური განხილვა იხილეთ ქვემოთ). მიუხედავად ამისა, როგორც წესი, საერთო მეხსიერების მქონე მრავალპროცესიან პროცესორში მაღალმწარმოებლური პარალელური გამოთვლებისათვის გამოიყენება პარალელური ალგორითმები, მაშინ როცა ფართომასშტაბიანი განაწილებული სისტემის კოორდინაციისათვის განაწილებული ალგორითმები გამოიყენება.



ნახ. 3. განაწილებული სისტემა: ა)ტიპიური განაწილებული სქემა, ბ) თანხვედრილი გ)პარალელური.

სხვადასხვა აპარატურისა და პროგრამული უზრუნველყოფის არქიტექტურა გამოიყენება განაწილებული გამოთვლებისათვის. შედარებით დაბალ დონეზე, საჭიროა რამდენიმე პროცესორის შეერთება რაიმე ქსელის მეშვეობით, იმის მიუხედავად ეს ქსელი იბეჭდება მიკროსქემის ბორტზე თუ შედგება სუსტად დაკავშირებული მოწყობილობებით და კაბელებით. უფრო მაღალ დონეზე აუცილებელია იმ პროცესების დაკავშირება, რომლებიც ამ პროცესორებზე სრულდება რაიმე საკომუნიკაციო სისტემით.

განაწილებული პროგრამირება, როგორც წესი, მიეკუთვნება რამდენიმე ძირითადი არქიტექტურიდან ერთ-ერთს: კლიენტ-სერვერულს, სამდონიანს, n- დონიანს, ან ერთრანგიანს; ან კიდევ კატეგორიებს: სუსტი კავშირი ან მჭიდრო კავშირი.

• კლიენტი-სერვერი: არქიტექტურა, სადაც ჭკვიანი კლიენტები უკავშირდებიან სერვერს მონაცემთა მისაღებად, ხოლო შემდეგ ახდენენ ფორმატირებას და აჩვენებენ მას მომხმარებლებს. კლიენტთან შესვლა ისევ ფიქსირდება სერვერზე, როდესაც ეს წარმოადგენს მუდმივ ცვლილებას.

• სამდონიანი: ეს არის არქიტექტურა, რომელიც გადაადგილებს კლიენტის ინტელექტს საშუალო დონეზე ისე, რომ მდგომარეობის შენახვის გარეშე (stateless) კლიენტები შეიძლება გამოყენებულ იქნან. ეს ამარტივებს აპლიკაციის განლაგებას. ვებ-აპლიკაციების უმრავლესობა სამდონიანია.

• n-დონიანი: არქიტექტურები, მიეკუთვნებიან ჩვეულებრივ ვებ-აპლიკაციებს, რომლებიც შემდგომში უგზავნიან თავიანთ მოთხოვნებს სხვა კორპორატიულ სამსახურებს. ამ ტიპის აპლიკაცია ყველაზე მეტადაა პასუხისმგებელი აპლიკაციების სერვერების წარმატებაზე.

• ერთრანგიანი: ეს არის არქიტექტურები, სადაც არ არის სპეციალური მანქანები, რომლებიც უზრუნველყოფენ მომსახურებას ან ქსელის რესურსების მართვას. ამის ნაცვლად ყველა პასუხისმგებლობა ერთნაირად იყოფა ყველა აპარატში, რომელიც ცნობილია, როგორც ერთრანგიანი კვანძები. ერთრანგიანებს შეუძლიათ იყვნენ როგორც კლიენტები, ისე სერვერები.

განაწილებული გამოთვლითი არქიტექტურის კიდევ ერთი ძირითადი ასპექტი არის, თანხვედრი პროცესების მუშაობის კომუნიკაციის და კოორდინაციის მეთოდი. სხვადასხვა გზავნილის გადაცემის პროტოკოლების საშუალებით პროცესები შეიძლება პირდაპირ დაუკავშირდეს ერთმანეთს, როგორც წესი, „ოსტატი“/„დაქვემდებარებული“ ურთიერთობებით. გარდა ამისა, არქიტექტურა "მონაცემთა ბაზის ცენტრი" განაწილებულ გამოთვლებს საშუალებას აძლევს იმუშონ პირდაპირი ინტერ-პროცესის კომუნიკაციის გარეშე, საერთო მონაცემთა ბაზის გამოყენებით.

განაწილებული სისტემებისა და განაწილებული გამოთვლების გამოყენების მიზეზები შეიძლება მოიცავდეს შემდეგს:

1. თვითონ განაცხადის ბუნებამ შეიძლება მოითხოვოს ისეთი საკომუნიკაციო ქსელის გამოყენება, რომელიც რამდენიმე კომპიუტერს აკავშირებს: მაგალითად, ერთ ფიზიკურ ადგილას წარმოიქმნილი მონაცემები, რომლებიც საჭიროა სხვა ადგილას.

2.არსებობს ბევრი შემთხვევა, როდესაც ერთი კომპიუტერის გამოყენება პრინციპში შესაძლებელია, მაგრამ განაწილებული სისტემის გამოყენება პრაქტიკული მოსაზრებით სასარგებლოა. მაგალითად, ეს შეიძლება იყოს ეკონომიკურად უფრო ეფექტური, მწარმოებლობის სასურველი დონის მიღწევა რამდენიმე დაბალდონიანი კომპიუტერული კლასტერის გამოყენებით, შედარებით მაღალი დონის ერთი გამოთვლის ნაცვლად.

განაწილებული სისტემა უფრო დიდ საიმედოობას უზრუნველყოფს, ვიდრე გაუნაწილებელი სისტემა, რადგან არ არსებობს ერთიანი მარცხი. უფრო მეტიც, განაწილებული სისტემის გაფართოება და მართვა უფრო ადვილია, ვიდრე მონოლითური ერთპროცესორული სისტემის.

განაწილებული სისტემებისა და განაწილებული კომპიუტერების გამოყენების მაგალითებია:

სატელეკომუნიკაციო ქსელები:

- სატელეფონო ქსელები და ფიჭური ქსელები;
- კომპიუტერული ქსელები, როგორიცაა ინტერნეტი;
- უკაბელო სენსორული ქსელები;
- მარშრუტიზაციის ალგორითმები.

ქსელის აპლიკაციები:

- მსოფლიო ქსელი და ერთრანგიანი ქსელები;
- მასობრივად მრავალგამოყენებადი ონლაინ თამაშები და ვირტუალური რეალობის ერთობა;
- მონაცემთა ბაზების გავრცელება და მონაცემთა ბაზების მართვის სისტემების გავრცელება;
- ქსელის ფაილური სისტემები;
- განაწილებული ინფორმაციის დამუშავების სისტემები, როგორიცაა საბანკო სისტემები და ავიაკომპანიის ბილეთების დაჯავშნის სისტემები;

რეალურ დროში პროცესის კონტროლი:

- საჰაერო კონტროლის სისტემები;
- სამრეწველო კონტროლის სისტემები.

პარალელური გამოთვლები:

- სამეცნიერო გამოთვლები, მათ შორის კლასეტური გამოთვლები და გრიდ-გამოთვლების ჩათვლით და სხვადასხვა მოხალისე კომპიუტერული პროექტები (იხ. განაწილებული კომპიუტერული პროექტების სია),
- განაწილებული რენდერინგი კომპიუტერულ გრაფიკაში.

მოდელები

ბევრი ამოცანა, რომელის ავტომატიზებაც გვსურს კომპიუტერის გამოყენებით, კითხვა-პასუხის ტიპისაა: ჩვენ კითხვის ვსვამთ და კომპიუტერმა უნდა წარმოადგინოს პასუხი. თეორიულ კომპიუტერულ მეცნიერებაში ასეთ ამოცანებს გამოთვლით ამოცანებს უწოდებენ. ფორმალურად, გამოთვლითი პრობლემა შედგება შემთხვევებისაგან, თითოეული ამ შემთხვევისათვის გამოსავალის მოძებნასთან ერთად. აქედან გამომდინარე, კითხვები არის ის რისი დასმაც შეგვიძლია და ამ კითხვებზე პასუხები კი - სასურველი.

თეორიული კომპიუტერული მეცნიერება ცდილობს გაიგოს, თუ რომელი კომპიუტერული პრობლემების გადაჭრაა შესაძლებელი კომპიუტერის (გამოთვლითი თეორიის) გამოყენებით და რამდენად ეფექტურად (გამოთვლითი სირთულის თეორია). ტრადიციულად, ნათქვამია, რომ პრობლემა შეიძლება მოგვარდეს კომპიუტერის გამოყენებით, თუ ჩვენ შეგვიძლია შევქმნათ ალგორითმი, რომელიც აწარმოებს სწორი გადაწყვეტილების მიღებას ნებისმიერ შემთხვევაში. ასეთი ალგორითმის რეალიზება შეიძლება განხორციელდეს კომპიუტერული პროგრამის სახით, რომლის გაშვება ხდება ზოგადი დანიშნულების კომპიუტერზე: პროგრამა კითხულობს პრობლემას ინსტალაციიდან, შეასრულებს გარკვეულ გამოთვლებს და შეიმუშავებს გამოსავალს. ისეთი ფორმალიზმები, როგორიცაა შემთხვევითი წვდომის მანქანები ან უნივერსალური ტურინგ მანქანები შეიძლება გამოყენებულ იქნას, როგორც ზოგადი დანიშნულების კომპიუტერის აბსტრაქტული მოდელები, რომელიც ასრულებენ ასეთ ალგორითმს.

პარალელურ და განაწილებულ გამოთვლით სფეროში შესწავლილი საკითხები მსგავსია იმ საკითხებისა, რომელიც შეისწავლება მრავალჯერადი კომპიუტერების ან კომპიუტერის, რომელებიც ახორციელებენ ინტერაქციის პროცესების ქსელს, შემთხვევაში: რომელი გამოთვლითი პრობლემის გადაჭრა შეიძლება ასეთი ქსელში და რამდენად ეფექტურად? თუმცა, ცხადია, არ არის ნათელი, თუ რას ნიშნავს "პრობლემის გადაჭრა" კონკურენტული ან განაწილებული სისტემის შემთხვევაში: მაგალითად, რა არის ალგორითმის შემმუშავებლის ამოცანა და რა არის თანმიმდევრული ან განაწილებული ზოგადი დანიშნულების კომპიუტერის ექვივალენტი?

ქვემოთ მოყვანილი დისკუსია მრავალ კომპიუტერზეა ფოკუსირებული, თუმცა ბევრი პრობლემა ერთ კომპიუტერზე გაშვებული პარალელური პროცესებისათვისაც მსგავსია.

ჩვეულებრივ გამოიყენება სამი თვალსაზრისი:

პარალელური ალგორითმები საერთო მეხსიერების მქონე მოდელში

• ყველა პროცესორს აქვს საერთო მეხსიერებასთან წვდომა. ალგორითმის შემმუშავებელი ირჩევს თითოეული პროცესორის მიერ შესრულებულ პროგრამას.

• ერთ-ერთი თეორიული მოდელი ეს არის *პარალელური შემთხვევითი შეღწევის მანქანები* (PRAM). თუმცა, მოდელი PRAM გულისხმობს, საერთო მეხსიერებაში შეღწევის სინქრონულ ხელმისაწვდომობას.

• საერთო მეხსიერების პროგრამები შეიძლება გავრცელდეს განაწილებულ სისტემებზე, თუ ძირითადი ოპერაციული სისტემა ახდენს კვანძებს შორის კომუნიკაციის ინკაპსულირებას და პრაქტიკულად აერთიანებს მეხსიერებას ყველა ცალკეულ სისტემაში.

• მოდელი, რომელიც უფრო ახლოსაა რეალურ მრავალპროცესიან მანქანების ქცევასთან და ითვალისწინებს მანქანის ინსტრუქციების გამოყენებას, როგორიცაა შეადარება გაცვლით (CAS), ასინქრონული საერთო მეხსიერების მოდელი. არსებობს ამ მოდელის შესახებ მრავალი ნაშრომი.

პარალელური ალგორითმები გზავნილის გადაცემის მოდელში

• ალგორითმის შემმუშავებელი ირჩევს ქსელის სტრუქტურას, ასევე პროგრამას, რომელსაც ახორციელებს თითოეული კომპიუტერი.

• გამოიყენება მოდელები, როგორიცაა ბულის სქემა (ლოგიკური სქემა) და დახარისხების ქსელები. ბულის სქემა შეიძლება ჩაითვალოს როგორც კომპიუტერული ქსელი: თითოეული გეითი არის კომპიუტერი, რომელზედაც გაშვებულია ძალიან მარტივი კომპიუტერული პროგრამა. იგივენაირად, დახარისხების ქსელი შეიძლება წარმოვიდგინოთ, როგორც კომპიუტერული ქსელი: თითოეული კომპარატორი არის კომპიუტერი.

განაწილებული ალგორითმები გაგზავნა-გადაცემის მოდელში

• ალგორითმის შემმუშავებელი მხოლოდ კომპიუტერულ პროგრამას ირჩევს. ყველა კომპიუტერი ერთი და იგივე პროგრამით მუშაობს. სისტემა სწორად უნდა მუშაობდეს ქსელის სტრუქტურის მიუხედავად.

• ჩვეულებრივად გამოყენებული მოდელი არის გრაფი ერთი სასრული მანქანით თითო კვანძზე.

განაწილებული ალგორითმების შემთხვევაში, გამოთვლითი პრობლემები, როგორც წესი, დაკავშირებულია გრაფებთან. ხშირად გრაფი, რომელიც აღწერს კომპიუტერის ქსელის სტრუქტურას, არის პრობლემის მაგალითი. ეს ილუსტრირებულია შემდეგ მაგალითში

მაგალითი

განვიხილოთ მოცემული G გრაფის შეფერილობის მოძებნის გამოთვლითი პრობლემა. სხვადასხვა ველმა შეიძლება გამოიყენოს შემდეგი მიდგომები:

ცენტრალიზებული ალგორითმები

- გრაფი G კოდირებულია როგორც სტრიქონები და სტრიქონი მოიცემა, როგორც კომპიუტერისათვის შემავალი მონაცემები. კომპიუტერული პროგრამა გრაფის შეფერილობა აღმოაჩენს და ახდენს შეფერილობის კოდირებას სტრიქონის სახით და გამოჰყავს შედეგი.

პარალელური ალგორითმები

• კვლავ გრაფი G კოდირებულია, როგორც სტრიქონი. თუმცა, მრავალჯერადი კომპიუტერები პარალელურად იმავე სტრიქონში შედიან. თითოეულ კომპიუტერს შეუძლია ფოკუსირება მოახდინოს გრაფის ერთი ნაწილზე და შექმნას ამ ნაწილისათვის შეფერილობა.

• ძირითადი ყურადღება ექცევა მაღალმწარმოებლურ გამოთვლებს, რომელიც იყენებს რამდენიმე პარალელური კომპიუტერის სიმძლავრეს.

განაწილებული ალგორითმები

• გრაფი G არის კომპიუტერული ქსელის სტრუქტურა. არსებობს ერთი კომპიუტერი G-ის თითოეული კვანძისათვის და G-ის თითოეული წიბოსთვის ერთი მაკავშირებელი არხი. თავდაპირველად, თითოეული კომპიუტერი მხოლოდ იცნობს მის უშუალო მეზობლებს G გრაფაში; კომპიუტერებმა უნდა გაუზიარონ შეტყობინებები ერთმანეთს, რათა გაიგონ მეტი G-ს სტრუქტურის შესახებ. თითოეულმა კომპიუტერმა უნდა გამოიმუშავოს საკუთარი შეფერილობა, როგორც შედეგი (output).

• მთავარი ყურადღება გამახვილებულია თვითნებურად განაწილებული სისტემის ფუნქციონირების კოორდინაციაზე.

მიუხედავად იმისა, რომ პარალელური ალგორითმების სფეროს გააჩნია განსხვავებული მიმართულება, ვიდრე განაწილებული ალგორითმების სფეროს, არსებობს მრავალი ურთიერთქმედება ამ ორ სფეროს შორის. მაგალითად, გრაფების გაფერადებისათვის კოული -

ვიშკინის ალგორითმი თავდაპირველად წარმოდგენილი იყო როგორც პარალელური ალგორითმი, მაგრამ იგივე ტექნიკა შეიძლება გამოყენებულ იქნეს უშუალოდ, როგორც პირდაპირი განაწილებული ალგორითმის სახით.

უფრო მეტიც, პარალელური ალგორითმი შეიძლება განხორციელდეს პარალელურ სისტემაში (საერთო მეხსიერების გამოყენებით) ან განაწილებული სისტემაში (გზავნილის გამოყენებით). ტრადიციული საზღვარი პარალელურ და განაწილებულ ალგორითმებს შორის (აირჩიეთ შესაფერისი ქსელი და სამუშაო ნებისმიერ ამ ქსელში) არ ემთხვევა პარალელური და განაწილებული სისტემებს შორის საზღვარს (საერთო მეხსიერება და შეტყობინების გაგზავნა).

სირთულეები

პარალელურ ალგორითმებში, დროისა და სივრცის გარდა არსებობს კიდევ ერთი რესურსი, კომპიუტერების რაოდენობა. სინამდვილეში, ხშირად არსებობს კომპრომისი მუშაობის დროსა და კომპიუტერების რაოდენობას შორის: პრობლემა შეიძლება გადაწყდეს უფრო სწრაფად, თუ არსებობს პარალელურად გაშვებული მეტი კომპიუტერი (იხ. დაჩქარება). თუ გადაწყვეტილების მიღება შესაძლებელია პოლილოგარითმულ დროში მოგვარდეს პროცესორების მრავალწევრიანი რიცხვის გამოყენებით, მაშინ ამბობენ, რომ პრობლემა NC კლასში იმყოფება. NC კლასი თანაბრად კარგად შეიძლება განისაზღვროს PRAM ფორმალიზმის გამოყენებით ან ლოგიკური სქემებით - PRAM მანქანებს ეფექტურად შეუძლიათ მოახდინონ ლოგიკური სქემების მოდელირება და პირიქით.

განაწილებული ალგორითმების ანალიზის დროს ჩვეულებრივ უფრო მეტი ყურადღება ექცევა საკომუნიკაციო ოპერაციებს, ვიდრე გამოთვლით ეტაპებს. ალბათ განაწილებული გამოთვლების უმარტივესი მოდელი არის სინქრონული სისტემა, სადაც ყველა კვანძის მუშაობა მყარი კონფიგურაციით მიდის. ეს მოდელი საყოველთაოდ ცნობილია როგორც ლოკალური მოდელი. თითოეული საკომუნიკაციო რაუნდის დროს, პარალელურად (1) ყველა კვანძი მიიღებს უახლეს შეტყობინებებს მეზობლებისგან, (2) განახორციელებს ნებისმიერ ადგილობრივი გამოთვლებს და (3) აგზავნის ახალ შეტყობინებებს მეზობლებთან. ასეთ სისტემებში, ძირითადი სირთულე არის სინქრონული კომუნიკაციის ის რაუნდების რაოდენობა, რომელმაც უნდა დაასრულოს ამოცანა.

ამ სირთულის ზომა მჭიდროდაა დაკავშირებული ქსელის დიამეტრთან. ვთქვათ D ქსელის დიამეტრია. ერთის მხრივ, ნებისმიერი გამოთვლითი პრობლემა შეიძლება გადაწყდეს ტრივიალურად სინქრონული დისტრიბუციის სისტემაში დაახლოებით 2D საკომუნიკაციო რაუნდში: უბრალოდ შეაგროვეთ ყველა ინფორმაცია ერთ ადგილას (D რაუნდები), გადაწყვიტეთ პრობლემა და შეატყობინეთ ყოველი კვანძის ამ გადაწყვეტილების შესახებ (D რაუნდები).

მეორეს მხრივ, თუ ალგორითმის შესრულების დრო გაცილებით მცირეა D რაოდენობის საკომუნიკაციო რაუნდებზე, მაშინ ქსელში კვანძებმა უნდა წარმოადგინონ თავისი გამომავალი მონაცემები ქსელის შორეული ნაწილის შესახებ ინფორმაციის მიღების შესაძლებლობის გარეშე. სხვა სიტყვებით რომ ვთქვათ, კვანძებმა უნდა მიიღონ გლობალურად შეთანხმებული გადაწყვეტილებები იმ ინფორმაციის საფუძველზე, რომლებიც ხელმისაწვდომია ადგილობრივ D-სამეზობლოში. მრავალი განაწილებული ალგორითმი ცნობილია შესრულების იმ დროზე ნაკლებით, ვიდრე D რაუნდია და იმის გაგება, თუ რომელი პრობლემის მოგვარებაა შესაძლებელია ასეთი ალგორითმებით ამ სფეროს კვლევის ერთ-ერთი ცენტრალური საკითხია. როგორც წესი, ალგორითმი, რომელიც წყვეტს პრობლემას ქსელის ზომაში პოლილოგარითმიურ დროში, ამ მოდელში ეფექტურად მიიჩნევა.

კიდევ ერთი საყოველთაოდ გამოყენებული ზომაა ქსელში (იხ. კომუნიკაციის სირთულე) გადაცემული ბიტების საერთო რაოდენობა. ამ კონცეფციის თვისებები, ჩვეულებრივ, CONGEST (B) მოდელით არის გამოსახული, რომელიც, როგორც წესი, განისაზღვრება LOCAL-ის მოდელით, მაგრამ სადაც ცალკეული შეტყობინებები შეიძლება შეიცავდეს მხოლოდ B ბიტს.

სხვა პრობლემები

ტრადიციული გამოთვლითი პრობლემები ვარაუდობენ, რომ ჩვენ ვსვავთ კითხვას, კომპიუტერი (ან განაწილებული სისტემა) გარკვეული დროის განმავლობაში ამუშავებს კითხვას, ხოლო შემდეგ იძლევა პასუხს და ჩერდება. ასეთი პრობლემების მაგალითებია მოსადილე ფილოსოფოსების პრობლემა და სხვა მსგავსი ურთიერთგამომრიცხავი პრობლემები. ამ პრობლემებში, განაწილებული სისტემა სავარაუდოდ კოორდინაციას გაუწევს საერთო რესურსების გამოყენებას ისე, რომ არ წარმოიქმნას კონფლიქტები ან ჩიხები.

ასევე არსებობს ფუნდამენტური გამოწვევები, რომლებიც უნიკალურია განაწილებული გამოთვლებისათვის. პირველი მაგალითია გამოწვევები, რომლებიც უკავშირდება გაუმართაობის მიმართ ტოლერანტობას. ამ პრობლემასთან დაკავშირებული მაგალითები მოიცავენ კონსენსუსის პრობლემას, ბიზანტიური გამტყუნების მიმართ ტოლერანტობას, და თვით-სტაბილიზაციისა.

ბევრი კვლევა ასევე ორიენტირებულია განაწილების სისტემების ასინქრონული ხასიათის გააზრებაზე:

• სინქრონიზატორები შეიძლება გამოყენებულ იქნას სინქრონული ალგორითმების გასაშვებად ასინქრონული სისტემებში.

• ლოგიკური საათები უზრუნველყოფს მიზეზობრივ-შედეგობრივ კავშირებს მოვლენებამდე.

• საათის სინქრონიზაციის ალგორითმები უზრუნველყოფს გლობალურად თანმიმდევრულად ფიზიკურ დროებს.

არჩევნები

კოორდინატორის არჩევნები (ან ლიდერის არჩევნები) - ეს არის ცალკეული პროცესის რაიმე ამოცანის, რომელიც განაწილებულია რამდენიმე კომპიუტერზე (კვანძზე), ორგანიზატორად დანიშვნის პროცესი. ამოცანის დაწყებამდე ქსელის ყველა კვანძმა ან არ იცის, თუ რომელი კვანძი იქნება "კოორდინატორი" (ან ლიდერი), ან ვერ ახერხებს დაუკავშირდნენ მიმდინარე კოორდინატორს. მაგრამ კოორდინატორის საარჩევნო ალგორითმის დასრულების შემდეგ, ყოველი კვანძი მთელს ქსელში აღიარებს კონკრეტულ, უნიკალურ კვანძს, ამოცანის კოორდინატორად.

ქსელის კვანძები ერთმანეთთან ურთიერთობენ, რათა გადაწყვიტონ, რომელი მათგანი შეასრულებს "კოორდინატორის" მოვალეობას. ამისათვის საჭიროა რაომელიმე მეთოდი იმისათვის, რომ დაირღვეს მათ შორის სიმეტრია. მაგალითად, თუ თითოეული კვანძის აქვს უნიკალური და შესადარებელი ინდეფიკატორი, მაშინ კვანძებს შეუძლიათ შეადარონ თავიანთი იდენფიკატორები და გადაწყვიტონ, რომ უმაღლესი იდენფიკატორის მქონე კვანძი კოორდინატორია.

ამ პრობლემის განსაზღვრას ხშირად მიაწერენ ლელანს (LeLann), რომელმაც ის ფორმალური გახადა, როგორც ტოკენ წრის იმ ქსელში ახალი ტოკენის შექმნის მეთოდი, რომელშიც ტოკენი იყო დაკარგული.

კოორდინატორის საარჩევნო ალგორითმები ისეა შემუშავებული, რომ ეკონომიური იყოს გადაცემული ბაიტების საერთო რაოდენობის და დროის თვალსაზრისით. გალაგერის, ჰუმბლეტისა და სპირის მიერ შემოთავაზებულმა ალგორითმმა ზოგადი არაორიენტირებული გრაფებისათვის დიდი გავლენა მოახდინა განაწილებული ალგორითმების შექმნაზე და ზოგადად მოიპოვა დეიკსტრის პრიზი (Dijkstra Prize), განაწილებული გამოთვლების შესახებ გავლენიანი სტატიის გამო.

მრავალი სხვა ალგორითმი იყო შემოთავაზებული სხვადასხვა სახის ქსელური გრაფებისთვის, როგორიცაა არამიმართული რგოლები, ერთიმიმართულებიანი რგოლები, სრული გრაფები, ბადეები, ეილერის მიმართული გრაფები, და სხვა. კორახმა, კუტენმა და მორანმა შემოგვთავაზეს ზოგადი მეთოდი, რომელიც გრაფის ოჯახის პრობლემას განაცალკევებს კოორდინატორის საარჩევნო ალგორითმის შექმნისაგან.

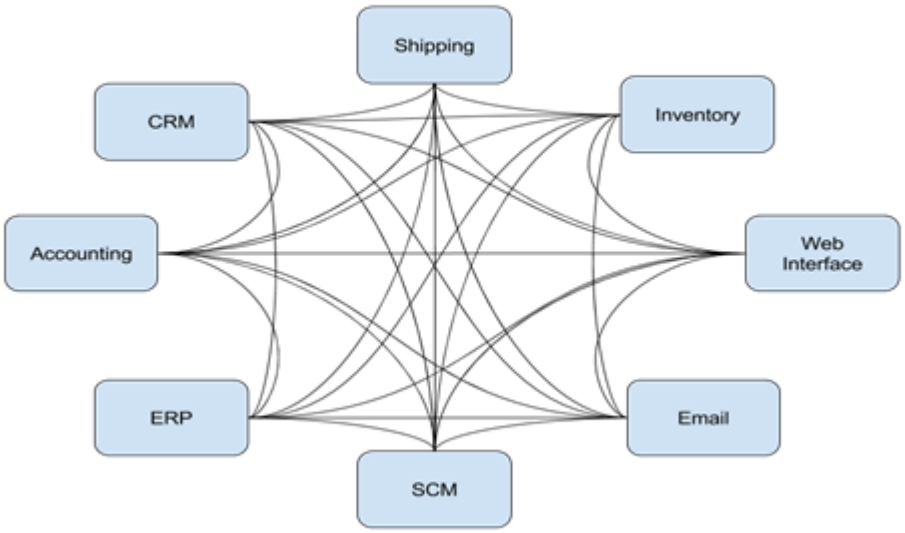
განაწილებული სისტემებში კოორდინაციის განსახორციელებლად კოორდინატორთა კონცეფციას იყენებენ. კოორდინატორის არჩევის პრობლემა მდგომარეობს იმაში, რომ განაწილებულ სისტემაში სხვადასხვა პროცესორებზე პროცესების ჯგუფიდან აირჩეს პროცესი, რათა მან იმოქმედოს, როგორც ცენტრალურმა კოორდინატორმა. არსებობს ცენტრალური კოორდინატორის არჩევის რამდენიმე ალგორითმები.

განაწილებული სისტემების თვისებები

აქამდე ყურადღებას ვამახვილებდით განაწილების იმ სისტემის შექმნაზე, რომელიც წყვეტს მოცემულ პრობლემას. კვლევის დამატებით პრობლემას წარმოადგენს მოცემული განაწილებული სისტემის თვისებების შესწავლა.

შეჩერების პრობლემა - ეს არის ანალოგიური მაგალითი ცენტრალიზებული გამოთვლითი სფეროდან: ჩვენ გვაქვს კომპიუტერული პროგრამა და პრობლემა მდგომარეობს შემდეგში, გადავწყვიტოთ ჩერდება ის თუ გრძელდება სამუდამოდ. ზოგადად შეჩერების პრობლემის გადაწყვეტა შეუძლებელია და კომპიუტერული ქსელის ქცევის ბუნებრივად გაგება, როგორც მინიმუმ, ისევე ძნელია, როგორც ერთი კომპიუტერის ქცევის გაგება.

თუმცა, არსებობს ბევრი საინტერესო განსაკუთრებული შემთხვევა, რომელის გადაწყვეტაც შესაძლებელია. კერძოდ, შეიძლება ვიმსჯელოთ, სასრული აპარატების ქსელის ქცევის შესახებ. ერთ-ერთი მაგალითი გვეუბნება შესაძლებელია თუ არა ურთიერთმოქმები ასეთი სასრული აპარატების ქსელის (ასინქრონული და არატრანზიციონალური) ჩიხში შესვლა. ამ პრობლემას წარმოადგენს PSPACE- ს სრული (PSPACE-complete) ვერსია, თუმცა მისი გადაწყვეტა შესაძლებელია, მაგრამ ნაკლებ სავარაუდოა, რომ არსებობს ეფექტური (ცენტრალიზებული, პარალელური ან განაწილებული) ალგორითმი, რომელიც პრობლემას წყვეტს დიდი ქსელების შემთხვევაში.[1]



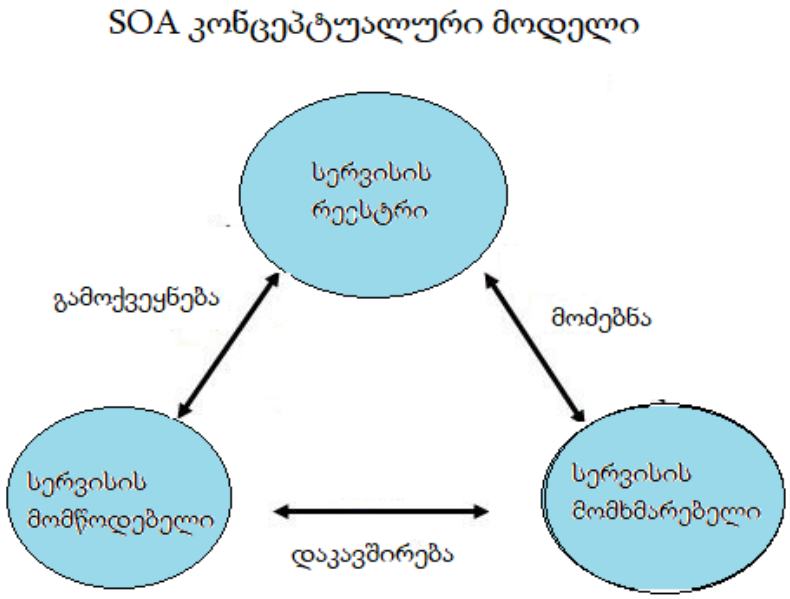
ნახ.4. „სპაგეტი“ ინტეგრაცია.
წყარო: https://dzone.com/articles/building-integration-solutions-a-rethink

განაწილებულმა სისტემებმა (უფრო ზუსტად, განაწილებული კომპიუტერული სისტემები) იმ ადგილიდან სადაც მოხდა მათი გაშვება გრძელი გზა გაიარეს. თავიდან ერთი კომპიუტერს შეეძლო ერთდროულად მხოლოდ ერთ კონკრეტულ ამოცანის შესრულება. თუ გვსურს შევასრულოთ რამდენიმე ამოცანა პარალელურად, ჩვენ დაგვჭირდება პარალელურად გაშვებული რამდენიმე კომპიუტერი. მაგრამ მათი პარალელურად გაშვება განაწილებული სისტემების აგებისათვის მართლაც არ არის საკმარისი, ის მოითხოვს სხვადასხვა კომპიუტერებს შორის (ან ამ კომპიუტერებზე გაშვებულ პროგრამებს შორის) კომუნიკაციისათვის მექანიზმს. რამდენიმე კომპიუტერს შორის მონაცემების გაცვლის

[1] https://en.wikipedia.org/wiki/Distributed_computing

(გაზიარების) ამ მოთხოვნამ გამოიწვია შეტყობინებაზე ორიენტირებული კომუნიკაციის იდეა, რომელიც ორიენტირებულია იმ შეტყობინებებზე, სადაც ორი კომპიუტერი ცვლის მონაცემებს: რომლებიც იყენებენ იმ შეტყობინებებს; რომლებიც ამუშავებენ მონაცემებს. არსებობდა რამდენიმე სხვა მექანიზმი, როგორიცაა ფაილის გაზიარება; მონაცემთა ბაზაში გაზიარება ასევე შევიდა ამ სურათში (ნახ. 4).

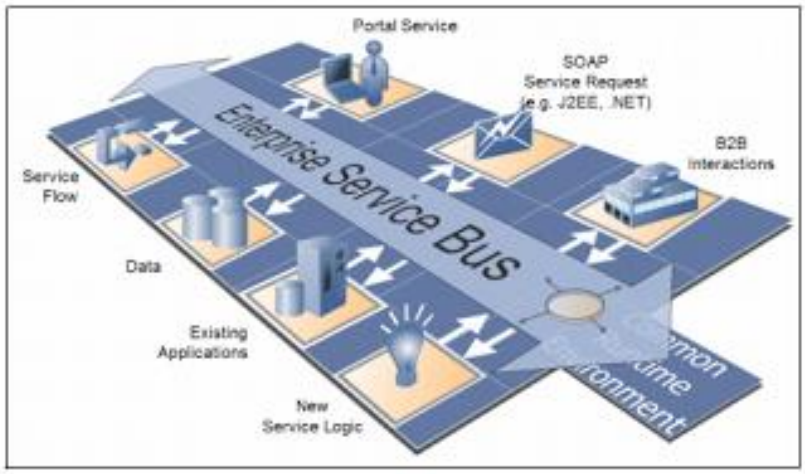
შემდეგ დადგა მრავალამოცანიანი ოპერაციული სისტემების და პერსონალური კომპიუტერების ერა. ოპერაციულ სისტემებში Windows, Unix და Linux შესაძლებელი იყო ერთ კომპიუტერზე რამდენიმე ამოცანის გაშვება. ამან შემმუშავებლებს საშუალება მისცა შეექმნათ და გაეშვათ განაწილებული სისტემა ერთ ან რამდენიმე კომპიუტერზე, რომლებიც ერთმანეთთან დაკავშირებულია შეტყობინებების გაცვლის საშუალებით. რამაც მიგვიყვანა სერვისზე ორიენტირებულ არქიტექტურამდე (SOA), სადაც თითოეული განაწილებული სისტემის აგება შესაძლებელი იყო იმ სერვისების კომპლექტით, რომელიც ერთ კომპიუტერზე ან მრავალ კომპიუტერზე მუშაობს. მომსახურების ინტერფეისი სწორად იყო განსაზღვრული WSDL (SOAP) ან WADL (REST- ისთვის) და მომსახურების მომხმარებლებმა გამოიყენეს ეს ინტერფეისი მათი კლიენტის მხარის (client-side) რეალიზებისათვის.



წყარო: https://www.akuaroworld.com/telecom-oss-new-network-architecture/soa-model/

გამოთვლით სიმძლავრეებზე და საცავებზე ფასების შემცირებასთან ერთად მთელს მსოფლიოში ორგანიზაციებმა დაიწყეს SOA-ს ბაზაზე დაფუძნებული განაწილებული სისტემების და კორპორატიული IT სისტემების გამოყენება. მას შემდეგ, რაც მომსახურების ან

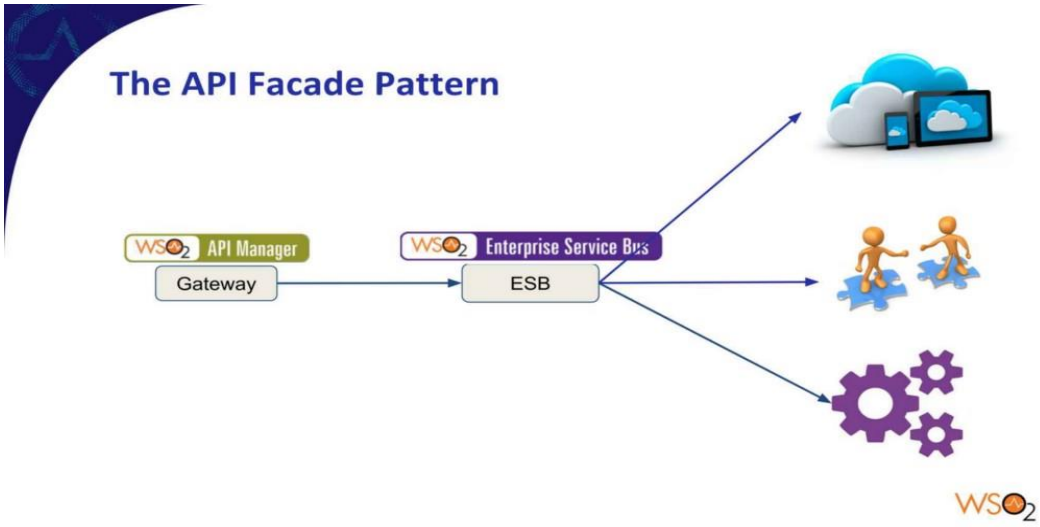
სისტემების რაოდენობა გაიზარდა, ამ სერვისების წერტილი-წერტილთან (point-to-point) კავშირი აღარ იყო სკალირებული და შენარჩუნებული. ეს იწვევს ცენტრალიზებულ "სერვის-სალტე (buss)" კონცეფციას, რომელიც აკავშირებს ყველა სხვადასხვა სისტემას, ჰაბის ტიპის არქიტექტურის მეშვეობით. ამ კომპონენტს ეწოდება ESB (საწარმოო სერვის-სალტე) და ის მოქმედებს, როგორც თარჯიმანი ან შუამავალი, იმ ადამიანებს შორის, რომლებიც საუბრობენ სხვადასხვა ენებზე, მაგრამ ერთმანეთთან კომუნიკაცია უნდათ. საწარმოში ენები იმ შეტყობინებების პროტოკოლების და მიმოწერის ფორმატების ანალოგიური იყო, რომელებსაც იყენებენ განსხვავებული სისტემები საკუთარი კომუნიკაციისათვის. ამ მოდელის მთავარი უპირატესობა იყო ის, რომ თითოეულ სისტემას შეუძლია შექმნას როგორც სერვერის მხარის ისე კლიენტის მხარის შესრულება, და არ იღელვოს დამაკავშირებელი სისტემების პროტოკოლების გამო.



წყარო: http://ibmwebsphereenterpriseservicebus.blogspot.com/2015/09/enterprise-requirements-for-esb.html

ეს მოდელი კარგად მუშაობდა და დღესაც კარგად მუშაობს. მსოფლიო ქსელის პოპულარობისა და მოდელის სიმარტივის გამო REST- ზე დაფუძნებული კომუნიკაცია უფრო პოპულარული გახდა, ვიდრე SOAP- ზე დაფუძნებული კომუნიკაციის მოდელი. ამან საფუძველი დაუდო აპლიკაციის პროგრამირების ინტერფეისსზე (API) დაფუძნებულ კომუნიკაციას REST მოდელის მეშვეობით. REST- ის მოდულის სიმარტივის გამო, ისეთი ფუნქციები, როგორიცაა უსაფრთხოება (აუტთენტიფიკაცია და ავტორიზაცია), ქეშირების, რეგულირება და მონიტორინგი ტიპის შესაძლებლობები სტანდარტული REST API- ის შესრულების განხორციელების მიზნით აუცილებელი გახდა. იმის ნაცვლად, რომ მოეხდინათ ამ შესაძლებლობების განხორციელება თითოეულ API- ში ცალკ-ცალკე, წარმოიშვა მოთხოვნა, რომ ჰქონოდათ API- ზე საერთო კომპონენტი ამ ფუნქციების გამოყენების შესახებ.

ამ მოთხოვნას მივყავართ API მართვის პლატფორმის ევოლუციამდე და დღეს ეს გახდა ნებისმიერი განაწილებული სისტემის ერთ-ერთი ძირითადი მახასიათებელი.

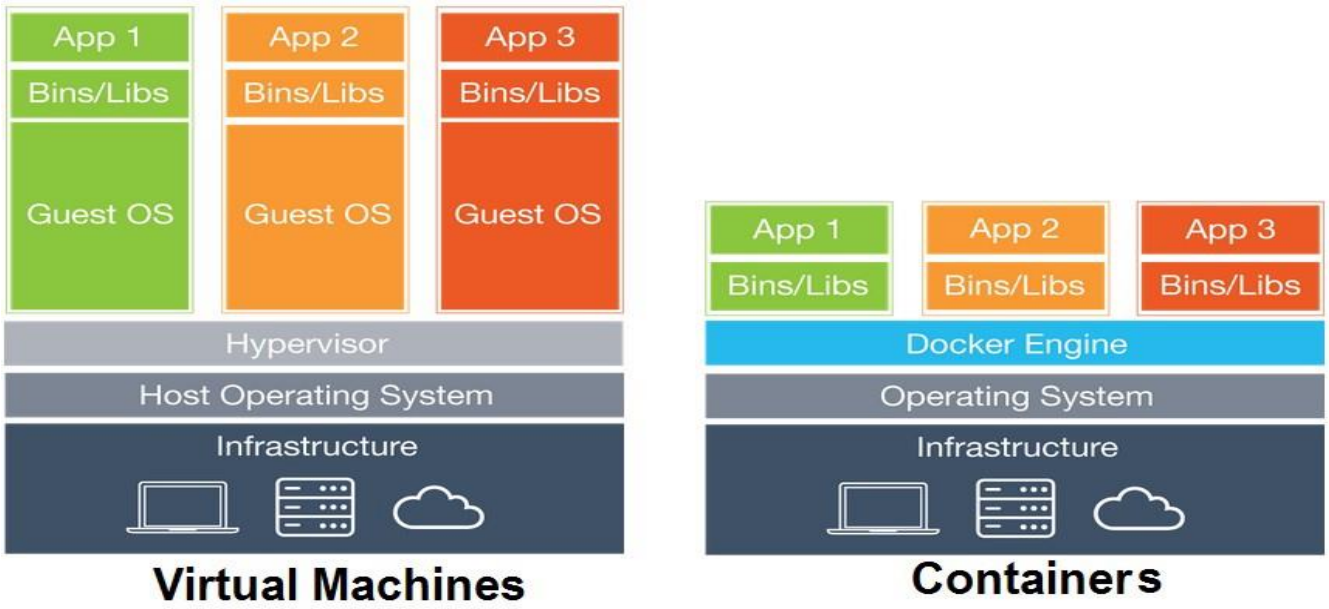


წყარო: https://www.infoq.com/presentations/api-management-architecture

შემდეგ, როდესაც ინტერნეტ კომპანიები, როგორებიცაა Facebook, Google, Amazon, Netflix, LinkedIn და Twitter, იმდენად გამსხვილდნენ, რომ მათ გაუჩნდათ სურვილი შეექმნათ ისეთი განაწილებული სისტემები, რომლებიც მოიცავდნენ რამდენიმე რეგიონს და მონაცემთა დამუშავების რამდენიმე ცენტრს, დადგა განაწილებული სისტემების დიდი აფეთქების მომენტი. ამ ტიპის მოთხოვნებმა გადაადგილა ტექნოლოგიების ფოკუსირება იმ ადგილისკენ, სადაც ეს ყველაფერი დაიწყო. თავიდან ინჟინერებმა დაიწყეს ფიქრი ერთიანი კომპიუტერის და ერთიანი პროგრამის კონცეფციაზე. იმის ნაცვლად, რომ ერთი კომპიუტერი განეხილათ, როგორც ერთი კომპიუტერი, ისინი ფიქრობენ იმაზე, თუ როგორ შეექმნათ რამდენიმე ვირტუალური კომპიუტერი ერთ მანქანაზე. ეს იწვევს ვირტუალური მანქანების შესახებ იდეის ლიდერობას, როდესაც ერთ კომპიუტერს შეუძლია იმუშაოს როგორც რამდენიმე კომპიუტერმა და ახდენს ყველა მათგანის პარალელურად გაშვებას. მიუხედავად იმისა, რომ ეს საკმარისად კარგი იდეა იყო, იგი არ აღმოჩნდა საუკეთესო ვარიანტი, როდესაც საქმე მიდგა ჰოსტ-კომპიუტერის რესურსების გამოყენებაზე. სხვადასხვა საოპერაციო სისტემების გაშვება საჭიროებს დამატებით რესურსებს, რომლებიც არ იყო მოთხოვნილი იმავე ოპერაციულ სისტემაში გაშვებისას.

ამან მიგვიყვანა კონტეინერების იდეებამდე, სადაც გაშვებულია რამდენიმე პროგრამა და მათი გამოყენება დამოკიდებულია ცალკეულ გაშვების დროზე და საჭიროების მიხედვით

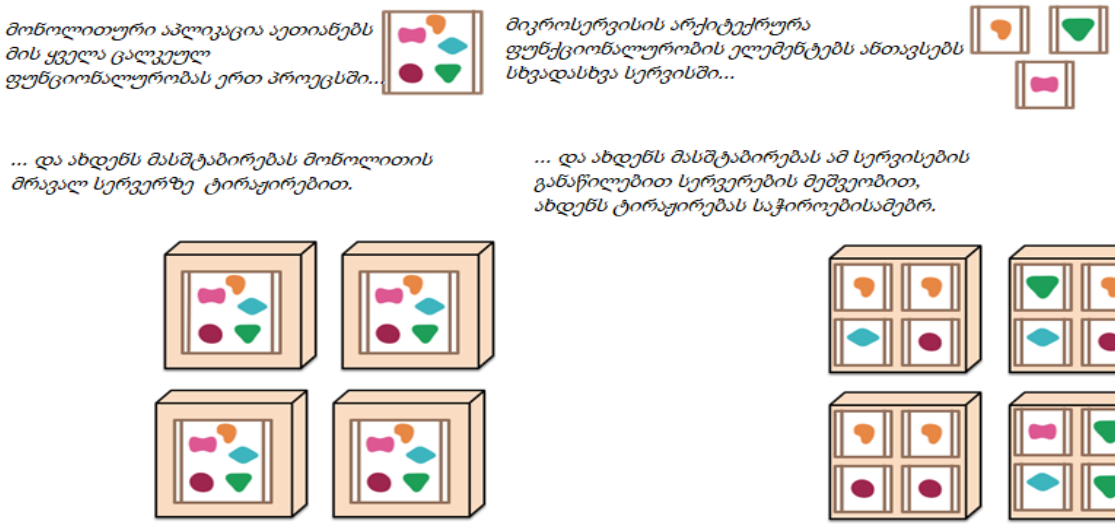
იყენებს იმავე ჰოსთ ოპერაციული სისტემის ბირთვს (Linux). ეს კონცეფცია ხელმისაწვდომი იყო Linux- ის ოპერაციულ სისტემაში გარკვეული დროის მანძილზე, ის უფრო პოპულარული ხდებოდა კონტეინერზე დაფუძნებული აპლიკაციების განლაგების დანერგვით. კონტეინერებმა შეიძლება იმოქმედონ ვირტუალური მანქანების მსგავსად ცალკე ოპერაციული სისტემა ოვერჰედის გარეშე. თქვენ შეგიძლიათ მოათავსით თქვენი აპლიკაცია და ყველა შესაბამისი დამოკიდებულებები კონტეინერის იმიჯში, რომელის გაშვებაც შეიძლება აწარმოოთ ნებისმიერ გარემოში, რომელსაც გააჩნია ჰოსტინგის ოპერაციული სისტემა და შეუძლია კონტეინერების გაშვება. Docker და Rocket არის კონტეინერების აგების 2 პოპულარული პლატფორმა.



წყარო: https://techglimpse.com/docker-installation-tutorial-centos/

ამან შეუქმნა საფუძველი ისეთ ორგანიზაციებს, როგორიცაა Netflix, LinkedIn, Twitter, რომ აეგოთ მუდმივმოთხოვნადი მრავალპროფილიანი აპლიკაციური პლატფორმა, რამდენიმე მონაცემთა დამუშავების ცენტრით. ეს არ მოხდა სირთულეების გარეშე. კონტეინერზე დაფუძნებული განლაგების მინიატურულმა ხასიათმა გამოიწვია პლატფორმის შენახვისა და ორკესტრირების სირთულე ზოგიერთი კონტეინერისათვის. მიკროსერვისის არქიტექტურის (MSA) გამოგონებით, მონოლითური აპლიკაცია იყოფა მიკროსერვრების მცირე მოცულობად, რომლებიც შეუძლიათ მთლიანი მომსახურების მოცემული ფუნქციონირების შესრულება და განთავსებულნი არიან კონტეინერებში (უმეტეს შემთხვევაში). ამას მოჰყვა განაწილებული სისტემების ეკოსისტემის მიმართ განსხვავებული მოთხოვნების კომპლექტი, რათა სისტემა

საბოლოოდ შეთანხმებული ყოფილიყო და ერთმანეთთან კომუნიკაცია დაემყარებინა ბევრი სირთულის გარეშე.

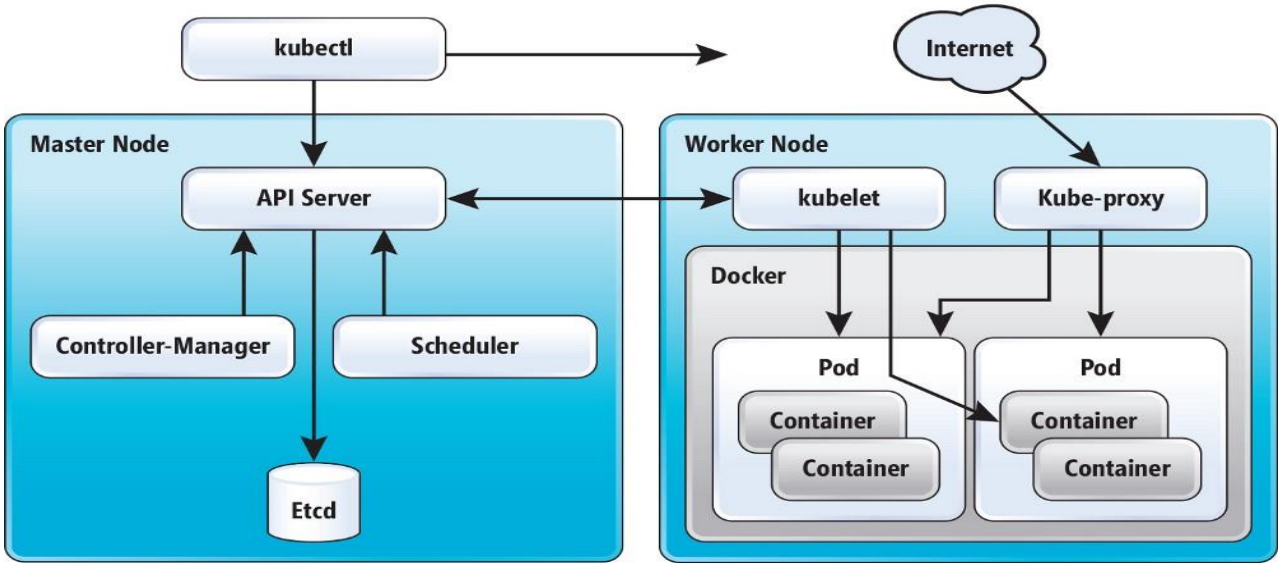


წყარო: https://martinfowler.com/articles/microservices.html

ეს მოთხოვნები საბოლოოდ დაეხმარა ინჟინერებს კონტეინერის ორკესტრირების სისტემის ჩამოყალიბებაში, რომელიც შეიძლება გამოყენებულ იქნას დიდი კონტეინერის ბაზაზე დისლოცირებული სისტემის მდგრადობის შენარჩუნების მიზნით. არ არის გასაკვირი, რომ მათი მასშტაბების გათვალისწინებით ამ დომენზე ცნობილი ტექნოლოგია Google-ისგან მოვიდა. მათ ააგეს საკონტეინერო ორკესტრირების პლატფორმა სახელწოდებით "კუბერნეტები" (a.k.a K8S) და ის გახდა ფართომასშტაბიანი კონტეინერის ორკესტრირების მოთხოვნებისთვის დეფაქტო სტანდარტი. K8S-მა ინჟინრებს საშუალება მისცა:

- მსხვილ კლასტერებში კონტეინერების გაშვების;
- განიხილონ მონაცემთა ცენტრი, როგორც ერთი კომპიუტერი;
- კომუნიკაცია სერვერებს შორის(რომლებიც კონტეინერებშია გაშვებული);
- ავტომატური მასშტაბირების, რამდენიმე სერვერს შორის დატვირთვის ბალანსირების.

კუბერნეტებმა და დოკერმა ცხოვრება გაუადვილეს აპლიკაციის პროგრამისტებს. ისინი აღარ დარდობდნენ იმაზე, თუ როგორ მოიქცევიან ისინი სხვადასხვა გარემოში (ოპერაციული სისტემები, DEV, ტესტები, PROD და ა.შ.), რადგან მათ მიერ აგებული კონტეინერის სახე პრაქტიკულად ერთნაირად იმუშავებს ყველა გარემოში იმის გათვალისწინებით, რომ ყველა დამოკიდებულებები მასშია შეფუთული.



წყარო: https://redmondmag.com/articles/2017/08/01/container-orchestration-with-kubernetes.aspx

მაგრამ მაინც, კონტეინურ და საორკესტრო ჩარჩოებთან ერთად უნდა არსებობდეს გუნდი, რომელიც მართავს ამ სერვერებს. რაც იმას ნიშნავს, რომ მონაცემთა დამუშავების ცენტრს სჭირდება ისეთი მართვის ტექნოლოგიები, როგორიცაა Docker და Kubernetes, რათა იყოს იმის გარანტია, რომ ის თავს გრძნობს როგორც აპლიკაციებისათვის ერთი კომპიუტერი. იმის ნაცვლად, ამას თქვენ აკეთებთ, როგორც როცა ამ ნაწილს ვინმე სხვა მართავს? ეს სწორედ ის არის, რაც ხდება სერვერების გარეშე არქიტექტურაში, სადაც თქვენს სერვერს მართავს მესამე მხარე Cloud პროვაიდერი, როგორიცაა Amazon (Lambda), Microsoft (Azure ფუნქციები) ან Google (Cloud ფუნქციები). ახლა განაწილებული სისტემის დაპროგრამირება აპლიკაციის შემმუშავებლების მიერ იქნება უზრუნველყოფილი, ხოლო ინფრასტრუქტურის მენეჯმენტი ღრუბლოვანი პროვაიდერის მიერ განხორციელდება. ეს არის განაწილებული სისტემების ევოლუციის უახლესი მდგომარეობა და ის განაგრძობს განვითარებას.[2]

[2] https://dzone.com/articles/the-evolution-of-distributed-systems

თემა 3. *ინტერნეტის საშუალებით მასშტაბური გამოთვლები – ქსელური სისტემებისათვის ტექნოლოგიები – თანამედროვე კომპიუტერებისათვის კლასტერები. ბლეიდ სერვერები*

„ღრუბლოვანი“ გამოთვლების პერსპექტივა გარდაუვალია, ამიტომ ამ ტექნოლოგიების ცოდნა აუცილებელია ნებისმიერი სპეციალისტისათვის, რომელიც თავისი მიმდინარე ან მომავალი საქმიანობით დაკავშირებულია თანამედროვე საინფორმაციო ტექნოლოგიებთან.

მოცემული თემის მიზანს წარმოადგენს გამოთვლითი ტექნიკის განვითარების ძირითადი ეტაპების გაცნობა. აპარატული უზრუნველყოფის განვითარების თანამედროვე ტენდენციების ანალიზი, რომლებსაც მივყავართ ღრუბლოვანი გამოთვლების ტექნოლოგიების წარმოშობამდე.

აპარატული უზრუნველყოფის განვითარება

იმისათვის რომ გავიგოთ, თუ როგორ გაჩნდა „ღრუბლოვანი“ გამოთვლა, აუცილებელია წარმოვადგინოთ გამოთვლის და გამოთვლითი ტექნიკის განვითარების პროცესის ძირითადი მომენტები.

ჩვენს დროში ცხოვრება კომპიუტერის გარეშე წარმოუდგენლად გვეჩვენება. გამოთვლითი ტექნიკის დანერგვამ თითქმის ყველა ცხოვრებისეულ ასპექტში შეაღწია, როგორც კერძო ისე პროფესიონალურში. კომპიუტერების განვითარების ტემპი საკმაოდ სწრაფი იყო. კომპიუტერების ევოლუციური განვითარების დასაწყისი 1939 წელი იყო, როდესაც შემუშავებული იქნა ორობითი არითმეტიკა და ის გახდა კომპიუტერული გამოთვლების და პროგრამირების ენის საფუძველი. 1939 წელს შექმნილი იყო ელექტროგამომთვლელი მანქანები, რომლებიც ახდენდნენ გამოთვლებს ციფრული ფორმით. გამომთვლელი მოწყობილობების გამოჩენა 1942 წელს ხდება, როდესაც გამოგონელი იყო მოწყობილობა, რომელსაც შეეძლო მექანიკურად რიცხვების შეკრება. გამოთვლა ხდებოდა ელექტრული ნათურების გამოყენებით.

1941 წელს ბერლინში გერმანულ ავიაციის ლაბორატორიაში გამოჩენილი კონრად ცუზეს მოდელი *Z3* იყო ერთ-ერთი მნიშვნელოვანი მოვლენა კომპიუტერების განვითარებაში, რადგან ეს მანქანა მხარს უჭერდა გაანგარიშებას როგორც მცურავი

წერტილით, ისე ორობითი არითმეტიკით. მოწყობილობა განიხილება, როგორც ყველაზე პირველი კომპიუტერი, რომელიც სრულად შრომისუნარიანი იყო. პროგრმირების ენა ითვლება „$Turing - complete$ “, თუ ის ხვდება იმავე გამომთვლელ კლასში, რომელშიც ტიურინგის მანქანაა.

თანამედროვე კომპიუტერების პირველი თაობა გამოჩნდა 1943 წელს, როდესაც შემუშავებული იქნა მარკ I და მანქანა კოლოსი. IBM ($International\ Busines\ Machiner\ Corporation$)-ისაგან ფინანსური მხარდაჭერით მარკი იქნა კონსტრუირებული და შემუშავებული ჰარვარდის უნივერსიტეტში. ეს იყო საერთო დანიშნულების ელექტრომექანიკური პროგრამირებადი კომპიუტერი. კომპიუტერების პირველი თაობა აგებული იქნა შემაერთებელი სადენების და ელექტრო ნათურების გამოყენებით (თერმოელექტრული ნათურების). მონაცემები ინახებოდა ქაღალდის პერფობარათებზე. კოლოსი გამოიყენებოდა მეორე მსოფლიო ომის დროს, დაშიფრული შეტყობინების გასაშიფრად.

გაშიფვრის ამოცანის შესასრულებლად, კოლოსი იყენებდა პერფობარათიდან მაღალ სიჩქარეზე წაკითხულ მონაცემთა ორ ნაკადს. კოლოსი აფასებდა მონაცემთა ნაკადს, ითვლიდა რა თითოეულ აღმოჩენილ დამთხვევას, რომელიც ეფუძნებოდა დაპროგრამირებად ლოგიკურ ფუნქციას. სხვა მონაცემებთან შედარებისათვის შექმნილი იყო ცალკე ნაკადი.

ამ ეპოქის სართო დანიშნულების კომპუტერი იყო $ENIAC$ (ელექტრონული რიცხვითი ინტეგრატორი და კომპიუტერი), რომელიც აგებული იქნა 1946 წელს. ის იყო პირველი კოპიუტერი რომელსაც შეეძლო გადაპროგრამირება იმისათვის, რომ გადაეწყვიტა გამოთვლილი პრობლემის სრული სპექტრი. $ENIAC$ შეიცავდა 18 000 თერმოელექტრულ ნათურას. ის 27 ტონაზე მეტს იწონიდა და მოიხმარდა საათში 25 კილოვატს ელექტროენერგიას. $ENIAC$ ასრულებდა 100 000 გამოთვლას წამში. ტრანზისტორის შექმნა ნიშნავდა, რომ არაეფექტური თერმოელექტრული ნათურები შეიძლებოდა შეცვლილიყო უფრო წვრილი და საიმედო კომპონენტებით. ეს იყო შემდგომი მთავარი ნაბიჯი გამოთვლების ისტორიაში.

$Transistorised$ კომპიუტერებმა საფუძველი დაუდეს კომპიუტერების მეორე თაობის გამოჩენას, რომლებიც დომინირებდნენ 1950 წლების ბოლოს და 1960-იანი წლების

დასაწყისში. ტრანზისტორების და დაბეჭდილი სქემების გამოყენების მიუხედავად ეს კომპიუტერები ჯერ კიდევ დიდი ზომის და ძვირადღირებული იყო. ძირითადად ისინი უნივერსიტეტების და მთავრობის მიერ გამოყენებოდა. ინტეგრალური სქემა ანუ ჩიპი შექმნილი იყო ჯეკ კილბის მიერ. ამ მიღწევისათვის 2000 წელს მან მიიღო ნობელის პრემია ფიზიკის დარგში.

კილბის გამოგონებამ გამოიწვია აფეთქება კოპიუტერების მესამე თაობის განვითარებაში. მიუხედავად იმისა, რომ პირველი ინტეგრალური სქემა შექმნილი იყო 1958 წელს, ჩიპები არ გამოიყენებოდა 1963 წლამდე. მეინფრეიმების ისტორიის ათვლა დაიწყო 1964 წელს უნივერსალური კომპიუტერული $IBMSystem/360$ სისტემის გამოჩენიდან, რომლის შემუშავებაზე IBM-მ 5 მილიარდი დოლარი დახარჯა.

მეინფრეიმი – ეს არის გამოთვლითი ცენტრის მთავარი კომპიუტერი დიდი მოცულობის შიგა და გარე მეხსიერებით. ის განკუთვნილია ისეთი ამოცანებისათვის, რომლებიც მოითხოვენ რთულ გამოთვლით ოპერაციებს. თვითონ ტერმინი „მეინფრეიმი“ წარმოშობილია ამ სისტემის ტიპიური პროცესორული სვეტების სახელწოდებისაგან. 1960-იან და 1980-იანი წლების დასაწყისში $System/360$ სისტემა უპირობო ლიდერი იყო ბაზარზე. მისი კლონების შეძენა ხდებოდა მრავალ ქვეყანაში. ამ დროისათვის ისეთმა მენიფრეიმებმა, როგორიც IBM360-ია გაზარდეს შენახვის და დამუშავების უნარი, ინტეგრალური სქემები მინიკომპიუტერების შემუშავების საშუალებას იძლეოდნენ, რაც მცირე კომპანიების დიდ რაოდენობას საშუალებას აძლევდა ეწარმოებინათ გამოთვლები. დიოდური სქემების მაღალი დონის ინტეგრაციამ მიგვიყვანა ძალიან პატარა გამოთვლითი ერთეულების განვითარებამდე, რამაც გამოიწვია გამოთვლების შემდეგი ნაბიჯის განვითარება.

1971 წლის ნოემბერში $Intel$-მა გამოუშვა მსოფლიოში პირველი მიკროპროცესორი $Intel\ 4004$. ეს იყო სრული ცენტრალური პროცესორი ერთ ჩიპზე და გახდა პირველი კომერციულად ხელმისაწვდომი მიკროპროცესორი. ეს შესაძლებელი გახდა სილიციუმის კონტროლის ელექტროდის ახალი ტექნოლოგიების განვითარების გამო. ამან ინჟინრებს შედარებით უფრო მეტი ტრანზიტორების გაერთიანების საშუალება მისცა იმ ჩიპზე, რომელიც შეასრულებდა გამოთვლებს დაბალ სიჩქარეებზე. ამ განვითარებამ ხელი შეუწყო მეოთხე თაობის კომპიუტერული პლატფორმის გაჩენას.

მეოთხე თაობის კომპიუტერები, რომლებიც ამ დროს ვითარდებოდნენ, იყენებდნენ მიკროპროცესორებს, რომლებიც იტევდნენ კომპიუტერული დამუშავების უნარს ერთადერთ ჩიპზე. ახდენდა რა $Intel$-ის მიერ შემუშავებული თავისუფალი შეღწევის მეხსიერების (RAM) კომბინირებას, მეოთხე თაობის კომპიუტერები უფრო ჩქარი იყო, როგორც არასდროს და იკავებდნენ გაცილებით ნაკლებ ფართობს. $Intel$ 4004 პროცესორებს შეეძლოდ შეესრულებინათ მხოლოდ 60 000 ინსტრუქცია წამში. მწარმოებლების მიერ ნაბდრთული მიკროპროცესორები, რომლებიც ვითარდებოდნენ $Intel$ 4004-ისაგან, გახდა პერსონალური კომპიუტერების განვითარების ბაზა, მცირე ზომის და საკმაოდ იაფი იმისათვის, რომ ფართო საზოგადოებისათვის ყოფილიყო ხელმისაწვდომი. პირველი კომერციულად ხელმისაწვდმი პერსონალური კომპიუტერი იყო 1974 წელის ბოლოს გამოშვებული $MITS\ Altair$ 88 . შემდგომში გამოშვებული იქნა ისეთი პერსონალური კომპიუტერები, როგორიცაა $Apple$ I და II, $Commodore\ PET$, $VIC-20$, $Commodore$ 64, და საბოლოოდ 1981 წელს ორიგინალური $IBM-PC$. PC-ის ერა სერიოზულად დაიწყო 1980-იანების შუაში. ამ დროის განმავლობაში $IBM-PC$, $Commodore\ Amiga$ და $Atari\ ST$ საზოგადოებისათვის ხელმისაწვდომ ყველაზე გავრცელებულ PC-ის პლატფორმებს წარმოადგენდნენ. იმისდა მიუხედავად, რომ $Intel$ 4004-ისაგან გამოგონებით დაწყებული მიკროგამომთვლელი სიმძლავრე და მეხსიერება ბევრჯერ გაძლიერდა, მაღალი დონის ინტეგრაციის ჩიპების ტექნოლოგია (LSI) ან ზემაღალი დონის ინტეგრაცია ($VLSI$) ძალიან არ შეცვლილა. ამიტომ ამჟამინდელი კომპიუტერების დიდი რაოდენობა ისევ ხვდება კომპიუტერების მეოთხე თაობის კატეგორიაში.

პერსონალური კომპიუტერების წარმოების მკვეთრ ზრდასთან ერთად 1990 წლების დასაწყისში დაიწყო მეინფრეიმების ბაზრის კრიზისი, რომლის პიკი 1993 წელზე მოდის. ბევრი ანალიტიკოსი ალაპარაკდა მეინფრეიმების სრულ გაქრობაზე, ინფორმაციის ცენტრალიზებული დამუშავებიდან (ორდონიანი „კლიენტი-სერვერი“ არქიტექტურით გაერთიანებული პერსონალური კომპიუტერების მეშვეობით) განაწილებაზე გადასვლაზე. ბევრმა მეინფრეიმები აღიქვა, როგორც გამოთვლითი ტექნიკის გუშინდელი დღე, თვლიდნენ რა რომ $Unix$- და PC- სერვერები უფრო თანამედროვე და პერსპექტიული იყო.

1994 წლიდან ისევ გაიზარდა მეინფრეიმების მიმართ ინტერესი. საქმე ისაა, რომ როგორც პრაქტიკამ გვიჩვენა, მეინფრეიმების საფუძველზე ცენტრალიზებული დამუშავება

წყვეტს საწარმოოს მასშტაბის ინფორმაციული სისტემის აგების ამოცანას უფრო იაფად, ვიდრე განაწილებადი. მრავალი იდეა, რომელიც ძევს ღრუბლოვანი გამოთვლების კონცეფციაში ასევე „გვაბრუნებს“, რა თქმა უნდა დროის მიხედვით შესწორებულ, მეინფრეიმების ეპოქაში. ბრუსტოლში HP–ის გამოკვლევის და შემუშავების ერთ-ერთმა წამყვანმა მეცნიერთანამშრომელმა ჯონ მეილიმ მიაქცია ყურადღება იმას, რომ *cloud computing* ძალიან გვაგონებსსხვა ტექნიკური დონის მეინფრეიმებს:„ყველაფერი მოდის მეინფრეიმებიდან. მეინფრეიმებმა გვასწავლეს, თუ როგორაა შესაძლებელი ერთ გარემოში აპლიკაციების იზოლირება – რომელიც კრიტიკულად საჭირო უნარია დღეს“.

თანამედროვე ინფრასტრუქტურული გადაწყვეტილებები

ყოველწლიურად იზრდება ბიზნესის მოთხოვნილებები მომსახურების მიწოდების უწყვეტობისადმი, ხოლო მოძველებულ მოწყობილობებზე უწყვეტი ფუნქციონირება პრაქტიკულად შეუძლებელია. *IT* -ვენდორები [3] აწარმოებენ და ნერგავენ უფრო მეტად ფუნქციონალურ და საიმედო აპარატულ და პროგრამულ გადაწყვეტილებებს. განვიხილოთ ინფრასტრუქტული გადაწყვეტილებების განვითარების ძირითადი ტენდენციები, რომლებიც ასეთუისე ხელს უწყობდნენ ღრუბლოვანი გამოთვლების კონცეფციის გაჩენას:

- კომპიუტერების მწარმოებლურობის ზრდა. მრავალპროცესორიანი და მრავალბირთვიანი გამოთვლითი სისტემის გაჩენა, **ბლეიდ-სისტემების განვითარება;**
- **მონაცემების შენახვის სისტემისა და ქსელის შექმნა**;
- ინფრასტრუქტურის კონსოლიდაცია. **ბლეიდ-სისტემის წარმოქნა.**

გამომთვლელი ტექნიკის საშუალებების განვითარების პროცესში ყოველთვის არსებობდა ამოცანების დიდი კლასი, რომელიც ითხოვდა მაღალი კონცენტრაციის გამოთვლით საშუალებებს. მათ შეიძლება მივაკუთვნოთ რთული რესურსტევადი გამოთვლები (სამეცნიერო ამოცანები, მათემატიკური მოდელირება), ასევე მრავალრიცხოვანი მომხმარებლის მომსახურების ამოცანები (მონაცემთა ბაზების განაწილება, ინტერნეტ-სერვისები, ჰოსტინგ).

[3] ვენდორი- გამყიდველი, მოვაჭრე.

არც ისე დიდიხნის წინ პროცესორების მწარმოებლებმა მიაღწიეს პროცესორის მიერ სიმძლავრის გამომუშავების გონივრულ შეზღუდვას, რომლისთვისაც მისი მწარმოებლურობა ძალიან მაღალია ფარდობითად დაბალ ღირებულებასთან. პროცესორების სიმძლავრის შემდგომი გაძლიერებისას აუცილებელი გახდა პროცესორების გასაგრილებლად არატრადიციული მეთოდის გამოყენება, რაც საკმაოდ მოუხერხებელი და ძვირია. აღმოჩნდა, რომ გამოთვლითი ცენტრის სიმძლავრის გასაზრდელად უფრო ეფექტურია ცალკეული გამოთვლითი მოდულების რაოდენობის და არა მწარმოებლურობის გაზრდა. ამან გამოიწვია მრავალპროცესორიანი და მოგვიანებით მრავალბირთვიანი გამოთვლითი სისტემის გაჩენა. ჩნდება მრავალპროცესორიანი სისტემები, რომლებიც მოიცავენ 4-ზე მეტ პროცესორს. მიმდინარე მომენტში არსებობენ პროცესორები 8 და მეტი ბირთვით, რომელთაგან თითოეული ეკვივალენტურია მწარმოებლურობის მიხედვით. იზრდება სლოტების რაოდენობა ოპერატიული მეხსიერების *მოდულების მისაერთებლად,* ასევე მათი ტევადობა და სიჩქარე.

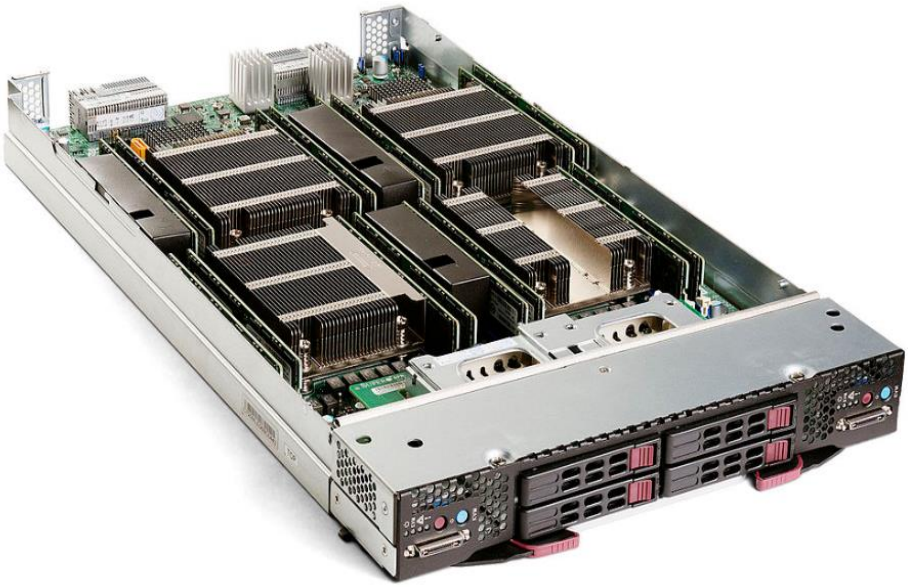
გამოთვლით ცენტრში გამოთვლითი მოდულების რაოდენობის გაზრდა მოითხოვს სერვერის განთავსების ახალ მიდგომებს, ასევე იწვევს ცენტრში მონაცემთა დამუშვების განთავსების, ელექტროკვების, გაგრილების და მომსახურების დანახარჯების ზრდას.

ამ პრობლემების გადასაჭრელად შეიქმნა XXI საუკუნის ახალი ტიპის სერვერები - მოდულური, ხშირად მოხსენიებული როგორც Blade-სერვერები. Blade-სერვერების უპირატესობას, რომლების პირველი მოდელი 2001 წელს შემუშავდა, მწარმოებლები აღწერენ წესი „1234“-ის მეშვეობით. „ღრუბლოვან სერვერებთან შედარებით შესადარისი მწარმოებლურობისას Blade-სერვერები იკავებენ ორჯერ ნაკლებ ადგილს, მოიხმარენ სამჯერ ნაკლებ ენერგიას და ოთხჯერ უფრო იაფი ჯდება“.

Blade სერვერი - ეს არის გაშიშვლებული მოდულური კონსტრუქციის მქონე სერვერული კომპიუტერი, რომელიც შექმნილია იმისთვის, რომ ოპტიმიზირებულად შეამციროს გამოყენების ფიზიკური სივრცე და ენერგია. Blade სერვერების ბევრი კომპონენტი ამოღებული აქვს, რომ მოხდეს სივრცის ეკონომია, ენერგიის და სხვა ფაქტორების მოხმარების მინიმუმამდე დაყვანა, ამასთან ყველა ფუნქციური კომპონენტები უნდა განიხილებოდეს როგორც კომპიუტერი. საკაბელო სერვერისგან განსხვავებით, ბლეიდის სერვერი საჭიროებს blade ჩანართებს, რომელსაც შეუძლია მრავალი blade სერვერის გამართვა,

ისეთი მომსახურება, როგორიცაა ენერგია, გაგრილება, ქსელი, სხვადასხვა ურთიერთკავშირი და მართვა. ერთად, blade და blade ჩანართი ქმნის blade სისტემას. სხვადასხვა blade პროვაიდერებს იმასთან დაკავშირებით, თუ რას უნდა შეიცავდეს blade თავად და blade სისტემა მთლიანად განსხვავებული პრინციპები აქვთ.

სტანდარტულ სერვერ-სტელაჟის კონფიგურაციაში, ერთი სტელაჟის ერთეული 1U-19 ინჩი (480 მმ) ფართო და 1.75 ინჩი (44 მმ) სიმაღლე - განსაზღვრავს ნებისმიერი აღჭურვილობის მინიმალურ ზომას. ბლეიდ-გამოთვლების ძირითადი სარგებელი და დასაბუთება ეხება ამ შეზღუდვის მოხსნას, რათა შეამციროს ზომის მოთხოვნები. ყველაზე გავრცელებული კომპიუტერის ფორმა-ფაქტორის სტელაჟებს გააჩნია 42U სამაღალე, რაც ზღუდავს უშუალოდ კომპიუტერის სტელაჟში ჩაშენებული დისკრეტული კომპიუტერის მოწყობილობების რაოდენობას 42 კომპონენტამდე. ბლეიდებს ეს შეზღუდვა არა აქვთ. 2014 წლის მდგომარეობით, blade სისტემის სიმკვრივემ მიაღწია 180 სერვერს ბლეიდ-სისტემაზე (ანუ სტელაჟზე 1440 სერვერს).



ნახ.5. Blade-სერვერი (SBI-7228R-T2X).

როდესაც საჭიროა შედარებით დაბალი ხარჯი და მეტი მწარმოებლობა, IT მენეჯერები ცდილობენ უფრო პატარა ფართში ჩატიონ მეტი გამოთვლითი სიმძლავრეები. ბლეიდ-სერვისების გამოყენებით კომპიუტერული ინფრასტრუქტურის განვითარება - ეს არის

ერთ-ერთი საშუალება. თუმცა, ამ რევოლუციურ მიდგომასაც აქვს თავისი დადებითი და უარყოფითი მხარეები.

რა უპირატესობა გააჩნია blade სერვერებს?

Blade სერვერები უზრუნველყოფს დიდ გამოთვლით სიმძლავრეს პატარა სივრცეში, რაც ამარტივებს კაბელების გაყვანას, შენახვას და მომსახურებას. Blade-ები ხშირად გამოიყენება *გრიდ-გამოთვლებისათვის*. ბლეიდ-სერვერების უპირატესობა მხოლოდ იმაში არ მდგომარეობს, რომ ერთ „კარადაში“ განთავსებულია რამდენიმე სერვერი, რომლებიც ერთად გამოიყენებენ კვების წყაროს და სხვა კომპონენტებს, არამედ დამაკავშირებელი რესურსების კონსოლიდაციაშიც, ისეთების როგორიცაა საცავები და ქსელური აღჭურვილობები, შენახვისა და ქსელის აპარატურა, შედარებით პატარა არქიტექტურაში, ვიდრე ეს იქნებოდა რეგულარული სერვერების ფერმის შემთხვევაში.

ჩატვირთვა დაბალანსება და წარუმატებლობა ...

ისევე, როგორც კლასტერიზაციის უმეტესობა აპლიკაციებში, ბლეიდ-სერვერების კიდევ ერთი უპირატესობა ის არის, რომ blade სერვერები შეიძლება ვმართოთ დატვირთვის დაბალანსების და გაუმართაობის დამუშავებისათვის. ეს ასევე შეიძლება გაკეთდეს ჩვეულებრივი სერვერების ფერმის მეშვეობით, მაგრამ რადგან ბლეიდ სერვერებს აქვთ ბევრად უფრო მარტივი და თხელი ინფრასტრუქტურა დატვირთვის დაბალანსების და გაუმართაობისადმი ტოლერანტობის მართვა უფრო ადვილია მათი საშუალებით.

ბლეიდ-სერვერზე ან თვითონ შასზე რაიმე აპარატული გაუმართაობის აღმოჩენის შემთხვევაში, ავტომატურად იწყებს თვით-დიაგნოსტიკის ფუნქციის გაშვებას და გაუმართავი ზონის იდენტიფიცირება ხდება ბლეიდ სერვერზე არსებული ინდიკატორის მეშვეობით.

ენერგომოხმარება და დენის წყაროს მართვა ...

ბლეიდ-სერვერი, მნიშვნელოვნად ამცირებს ენერგომოხმარებას და აუმჯობესებს დენის წყაროს მართვას. ბლეიდ-კორპუსში კვების ბლოკების კონსოლიდაცია ამცირებს საჭირო კვების ბლოკების რაოდენობას, ასევე ამცირებს სერვერზე კვების ბლოკების მიმართ მოთხოვნას. რამდენადაც ცალკეული ბლეიდ-სერვერები მინიმუმამდეა დაყვანილი და არ

გააჩნიათ სხვა ფუნქციები, რომლებიც გააჩნიათ ჩვეულებრივ სერვერებს, როგორიცაა კლავიატურა, გრაფიკული რუკები და სხვა, ამიტომ გამოიყენებენ ნაკლებ მოწყობილობებს, რომელთაც ესჭიროებათ კვება. ეს ამცირებს მთლიან ენერგომოხმარებას. ერთი-სერვერი 16 ბლეიდ სერვერით იყენებს ბევრად ნაკლებ ენერგიას, ვიდრე 16 ინდივიდუალური სრული ზომის სერვერი.

მართვის უფრო დაბალი ღირებულება ...

კიდევ ერთი დიდი უპირატესობა ტრადიციულ სერვერებთან შედარებით. ბლეიდ-სერვერები ყენდება ერთი ინტერფეისით, რომელიც გამოიყენება შასის ყველა ინდივიდუალური სერვერის სამართავად. სერვერების კონსოლიდაცია და რესურსების ცენტრალიზაცია ასევე ამარტივებს სერვერების განლაგებას, მართვასა და ადმინისტრირებას.

აპარატურის კონფიგურაციის მენეჯმენტი, ოპერაციის სტატუსის მონიტორინგი და გაუმართაობის მონიტორინგი ცენტრალიზებულია და ამცირებს სისტემის ადმინისტრატორის ტვირთს. ადმინისტრატორს შეიძლება დროულად ეცნობოს შეცდომის შესახებ ისეთი საშუალებებით, როგორცაა ელექტრონული ფოსტა.

საჭიროების შემთხვევაში, ადმინისტრატორს შეუძლია გაუშვას უკვე მომზადებული დისკის გამოსახულებები (OS და აპლიკაციების პროგრამული უზრუნველყოფა) ერთდროულად. ეს შესაძლებლობა რადიკალურად ამცირებს სისტემური ადმინისტრაციისთვის საჭირო დროს.

ქსელი და სხვა კაბელების გაყვანა ...

წარმოიდგინეთ, რომ თქვენ გაქვთ 16 ცალკეული სერვერი. თითოეული მათგანი უნდა იყოს დაკავშირებული ქსელთან, თითოეული ითხოვს იარდ ან მილ კაბელს. ბლეიდ სერვერები ამარტივებენ კაბელების მიმართ მოთხოვნებს და ამცირებენ სადენების რაოდენობას. ძალოვანი კაბელი, ოპერატორის გაყვანილობა (კლავიატურა, მაუსი და ა.შ.) და საკომუნიკაციო კაბელები (Ethernet, SAN კავშირები, კლასტერული კავშირი) მნიშვნელოვნადაა შემცირებული.

მოქნილობა, მოდულურობა და მარტივად განახლება ...

თანამედროვე ბლეიდ სერვერები ისეა დაპროექტებული, რომ შესაძლებელია ამოიღოთ და დაამატოთ ბლეიდები სანამ სისტემა მუშაობს და არის გაშვებული მხოლოდ მცირე

კონფიგურაციით ადმინისტრატორის ინტერფეისში. ახალი პროცესორის, კომუნიკაციის, შენახვისა და ურთიერთკავშირის ტექნოლოგიების განხორციელება შესაძლებელია ბლეიდებში, რომლებიც ინსტალირდება არსებულ აღჭურვილობებში მთლიანი სისტემის ფუნქციონალურობის მინიმალური დარღვევით. თქვენ შეგიძლიათ გააფართოვოთ ან შეცვალოთ სისტემის კონფიგურაცია დავალების შეწყვეტის გარეშე, იმ სერვერების გარდა, რომელებიც იმ ბლეიდ-სერვერებს მიეკუთვნებიან, რომელთა განახლება ან შეცვლაა საჭირო.

მოდულები შეიძლება შეერიოს იმავე ბლეიდ - სერვერის შასიზე. შასის შიგნით არაა აუცილებელი ბლეიდ-სერვერები იდენტურები იყვნენ. თქვენი შასის კონკრეტული ტიპის მიხედვით, ერთ სერვერს შეიძლება ჰქონდეს მაგალითად Intel Xeon და სხვას Intel Itanium. თქვენ შეგიძლიათ ერთი სერვერი გქონდეთ Windows- თან და მეორე Linux- თან.

განლაგება და სკალარულობა ...

იგივე ეხება სისტემაში ახალი სერვერის ბლეიდების დამატებას. ბლეიდ-სერვერის შასისი დაყენების შემდეგ შეიძლება დავუმატოთ ახალი სერვერები, მათი დამატებითი პლატფორმების უბრალოდ შეცურებით, სანამ სისტემა მთლიანად არის გაშვებული და მუშაობს.

საჭიროების შემთხვევაში შესაძლებელია ბლეიდების მასშტაბირება, უბრალოდ ახალი ბლეიდ-სერვერის დამატებით. რა თქმა უნდა, უფრო ადვილია ბლეიდ-სერვერის დამატება, ვიდრე ახალი სერვერის.

კატასტროფების მართვა ...

მიუხედავად იმისა, რომ ერთი შასში დამონტაჟებული ბლეიდ-სერვერების ერთობლიობა მიზნად ისახავს რესურსების ერთობლივ მოხმარებას, შასი შეიძლება კონფიგურირებული იყოს გადაჭარბებული სიმძლავრის მოდულით.

რა არის უარყოფითი მხარეები?

როგორც არანაირი სარგებელი არ მოდის უფასოდ - ბლედებში ასევე არის გარკვეული ნაკლოვანებები.

ძვირი კონფიგურაცია ...

თუმცა სისტემაში ახალი ბლეიდ-სერვერის ჩართვა ხდება მაშინ, როცა სისტემა მუშაობს, თავდაპირველი კონფიგურაცია შეიძლება იყოს შრომის ინტენსიური და ძვირადღირებული კომპლექსის გამოყენების ნაყოფი. ეს ნაკლი იმასთანაა დაკავშირებული, რომ ბლეიდ

სერვერები წარმოადგენენ სპეციალიზებულ გამოთვლით მოწყობილობებს, მათი კონფიგურაცია და ადმინისტრაცია ხშირად მოითხოვს გამყიდველის მიერ სწავლებას, რომელიც არ შეიძლება იყოს იაფი, თუ თქვენ არ გაქვთ გამყიდველთან სპეციალურ უფასო ტრენინგზე შეთანხმება.

ძვირადღირებული ინსტრუმენტი, ანუ მასშტაბებისაგან ეკონომია...

თუ არ შეავსებთ ბლეიდ-შასის სერვერული ბლეიდებით, თქვენ არ იყენებთ მას სრულად. აზრი არა აქვს შეიძინოთ ბლეიდ-შასი $ 5,000-ად და შემდეგ გაუშვათ მთელი სისტემა მხოლოდ 2 ბლეიდ-სერვერით. ბლედის შასი ხშირად გამოიყენება 14 ან 16 ბლეიდ-სერვერის განსათავსებლად.

ზოგადი წესი ის არის, რომ ბლეიდ სერვერები არ არის შესაფერისი და ეკონომიური იმ აპლიკაციებისათვის, რომლებიც მოითხოვენ 5-10 სერვერზე ნაკლებს. აპლიკაციები, რომლებიც მოითხოვენ 5-10 ნაკლებ სერვერულ ბლეიდებს (10-20 CPU), სრულად გამოსადეგია ავტონომიური სერვერის სისტემებისათვის.

გამყიდველი-საკეტი...

Blade სისტემები განსხვავდება სხვადასხვა მწარმოებლებელთან. მას შემდეგ, რაც თქვენ ხარჯავთ $ 50,000 ბლეიდ-სერვერის კონკრეტულ გამყიდველთან ყოველთვის არ არის ადვილი გადახვიდეთ სხვა გამყიდველთან, მომსახურებაზე ხელშეკრულებების და ასევე იმის გამო, რომ კონკურენტს, ნაკლებად სავარაუდოა იგივე გამოცდილება ჰქონდეს თქვენს ტექნიკაში, როგორც თქვენს გამყიდველს.

თეორიულად თქვენ შეგიძლიათ თქვენ გამოიყენოთ ბლედი-სერვერები კონკურენტის ბლეიდ-შასში, მაგრამ პრაქტიკულად ბლეიდ-შასები არ არიან სტანდარტიზებული. ნაკლებად სავარაუდოა, რომ IBM Dell და HP-სთან ერთად აიწყობა შასის ეთობლივად გამოსაყენებლად. შასი ეს არის ის, რაც მათ პროდუქტს უნიკალურს ხდის. ბლეიდ-სერვერი ხშირად მხოლოდ საკუთარი კომპანიის შასში მუშაობს.

ბიზნეს ქეისი ...

ბლეიდ სერვერები არ არის ყველაფერიდან საუკეთესო გამოსავალი. თუ თქვენ გაქვთ ტრანზაქციების დამუშავებისათვის ძალიან დიდი აპლიკაცია, რომელიც მოითხოვს მაღალი წაკითხვის/ ჩაწერის კოეფიციენტებს, მაშინ თქვენ მიადგებით ბოთლს ვიწრო ყელს ავტობუსის

სიჩქარით, მეხსიერების შეზღუდვების, დისკის ხელმისაწვდომობის და ქსელის I / O-ის გამო. ელ-ფოსტა და ვებ-მომსახურება ემსახურება სიტუაციებს, სადაც ბლეიდ-გამოთვლები კარგად არის შედგენილი.

გათბობა და გაგრილება ...

ერთი ხშირად დავიწყებული მინუსი არის HVAC. მაშინ, როცა რომ ინდივიდუალური ცალკე სერვერები შეიძლება გადანაწილდეს მთელს შენობაში და არ საჭიროებდნენ სპეციალურ დანადგარებს გაგრილებისათვის, ბლეიდ-სერვერები რომლებიც ამჟამად ძალიან ძლიერები არიან გამოყოფენ ძალიან დიდი რაოდენობის სითბოს თითო კვადრატულ ფუტზე. გაუფრთხილებლობის შემთხვევაში, შეიძლება მათი დადნობა. ბლეიდ-სერვერების შეძენისას მნიშვნელოვანია გვახსოვდეს, რომ HVAC-ისთვის საჭირო იქნება დამატებითი რესურსები.

განვიხილოთ ბლეიდ-სისტემის ძირითადი უპირატესობანი:

უნიკალური ფიზიკური კონსტრუქცია. ბლეიდ-სისტემის არქიტექტურა ემყარება დეტალურად დამუშავებულ უნიკალურ ფიზიკურ კონსტრუქციებს. ისეთი რესურსების ერთობლივი გამოყენება, როგორიცაა კვების საშუალება, გაგრილება, კომუტაციები და მართვა ამცირებს სირთულეებს და ახდენს იმ პრობლემების ლიკვიდაციას, რომლებიც დამახასიათებელია უფრო ტრადიციული სვეტური სერვერული ინფრასტუქტურებისათვის. ბლეიდ სისტემების ფიზიკური კონსტრუქცია გვთავაზობს ბლეიდ სერვერების განთავსებას სპეციალურ შასეში და მის ძირითად კონსტრუქტიულ ელემენტს წარმოადგენს გაერთიანებული პანელი. გაერთიანებული პანელი ისეა შემუშავებული, რომ ის წყვეტს ბლეიდ სერვერების კომუნიკაციის ყველა ამოცანას გარე სამყაროსთან: $Ethernet$ ქსელებთან, მონაცემთა შენახვის $Fiber\ Channel$ ქსელებთან, ასევე უზრუნველყოფს $SAS(SCSI)$ პროტოკოლის მიხედვით ურთიერთმოქმედებას იმავე *შასში* დისკურ ქვესისტემებთან. ბლეიდებისათვის *შასები* ასევე საშუალებას იძლევიან მათში განვათავსოთ გარე ქსელებთან კავშირისათვის აუცილებელი კომუტატორები $Ethernet$ ან $Fiber\ Channel$. ბლეიდ სერვერებიდან ამ კომუტატორებზე გამოსვლას უზრუნველყოფენ წინასწარ დაყენებული ან დამატებით დაყენებული კონტროლერები. საერთო თაროში ინტეგრირებული გარე სამყაროსთან კომუნიკაციის საშუალებები მნიშვნელოვნად ამცირებენ LAN და SAN-თან მისაერთებელად საჭირო კაბელების რაოდენობას ტრადიციულ სვეტურ სერვერებთან შედარებით. ბლეიდ სერვერებს გააჩნიათ საერთო კვება და გაგრილება. ცალკეულ სერვერებზე

კი არა, საერთო თაროზე კვების და გაგრილების სისტემების განლაგება უზრუნველყოფს ენერგომოხმარების დაწევას და საიმედოობის გაზრდას.

ნახ. 6. ტიპიური 10*U* კომპიუტერის სალტე10 Blade -სერვერისათვის (*Sun blade* 6000). http://imexresearch.com/newsletters/sunblade6000c7-17.htm

მართვის უკეთესი შესაძლებლობები და მოქნილობა. ბლეიდ-სერვერები პრინციპულად განსხვავდებიან სვეტურ სერვერებისაგან იმით, რომ სერვეულ თაროს აქვს ინტელექტი მოდულების მართვს სახით, რომლებიც არაა სვეტებში ტრადიციული სერვერების განთავსებისას. სისტემის მართვისთვის არაა აუცილებელი კლავიატურა, ვიდეო და მაუსი. ბლეიდ სისტემის მართვა ხდება მართვის ცენტრალიზებული მოდულის და თითოეულ ბლეიდ-სერვერზე დისტანციური მართვის სპეციალური პროცესორის მეშვეობით. *შასის* და სერვერების მართვის სისტემას, როგორც წესი, აქვს მართვისათვის საკმარისად მოხერხებული პროგრამული უზრუნველყოფა. ჩნდება მთლიანი „Blade“-სისტემის დისტანციური მართვის შესაძლებლობები, მათ შორის ელექტროკვებით და ცალკეული კვანძებით მართვა.

სკალაცია- საწარმოო სიმძლავრის მომატების აუცილებლობის შემთხვევაში, საკმარისია შევიძინოთ დამატებითი ბლეიდები და მივუერთოთ *შასს.* ბლეიდ-სისტემაში სერვერებს და ინფრასტრუქტურულ ელემენტებს ნაკლები ზომა აქვთ და მცირე ადგილს იკავებენ, ვიდრე ანალოგიური სვეტური გადაწყვეტილებანი, რაც გვეხმარება დავზოგოთ ელექტროენერგია და სივრცე, რომელიც *IT*-სთვის არის გამოყოფილი. გარდა ამისა, მოდულური აქრქიტექტურის წყალობით, ისინი უფორ მოხერხებული არიან დანერგვის და მოდერნიზაციის კუთხით.

გაზრდილი საიმედოობა. ტრადიციულ სვეტურ გარემოში საიმედოობის გაზრდისათვის მონტაჟდება დამატებითი მოწყობილობები, კომუნიკაციის საშუალებები და ქსელური კომპონენტები, რომლებიც უზრუნველყოფენ რეზერვირებას, რაც იწვევეს დამატებით ხარჯებს. ბლეიდ-სისტემებს აქვთ ჩადგმული რეზრევირების საშუალებები, მაგალითად გათვალისწინებულია კვების რამდენიმე ბლოკის არსებობა, რაც საშუალებას იძლევა კვების ერთი ბლოკის მწყობრიდან გამოსვლის შემთხვევაში უზრუნველვყოთ შასში განლაგებული ყველა სერვერის უწყვეტი მუშაობა. ასევე ხდება გამაგრილებელი კომპონენტების დუბლირება. ერთ-ერთი ვენტილატორის მწყობრიდან გამოსვლა არ იწვევს კრიტიკულ შედეგებს. ერთ-ერთი სერვერის მწყობრიდან გამოსვლის დროს სისტემური ადმინისტრატორი უბრალოდ ცვლის ბლეიდებს ახლით, ხოლო შემდეგ დისტანციურ რეჟიმში ახდენს მასზე ოპერაციული სისტემის და გამოყენებითი პროგრამული უზრუნველყოფის ინსტალაციას.

საექსპლუატაციო ხარჯების დაწევა. ბლეიდ-არქიტექტურის გამოყენებას მივყავართ ელექტრომოხმარების და სითბოს გამოყოფის შემცირებასთან, ასევე ამცირებს დაკავებულ მოცულობას. მონაცემთა დამუშავების ცენტრში დაკავებული ფართის შემცირებასთან ერთად, ბლეიდებზე გადასვლის ეკონომიკური ეფექტს გააჩნია კიდევ რამდენიმე შემადგენელი კომპონენტი. რამდენდაც მათში შედის ნაკლები კომპონენტი, ვიდრე ჩვეულებრივ სვეტურ სერვერებში, ისინი ხშირად იყენებენ დაბალვოლტიანი მოდელის პროცესორებს, რაც ამცირებს ელექტრომომარაგებაზე და გამაგრილებელ მანქანებზე მოთხოვნას. ბლეიდ-სისტემების ინფრასტუქტურა წარმოადგენს უფრო მარტივად სამართავს, ვიდრე სვეტურ სერვერებზე ტრადიციული IT- ინფრასტუქტურები. ზოგიერთ შემთხვევაში ბლეიდ-სისტემები კომპანიებს საშუალებას აძლევს ორჯერზე უფრო მეტად გაზარდონ რესურსების რაოდენობა ერთი ადმინისტრატორის მართვის ქვეშ (სერვერები, კომუტატორები და შენახვის სისტემები). მმართველი პროგრამული უზრუნველყოფა IT-ორგანიზაცებს ბლეიდ-სისტემების ეფექტური გაშლის, მონიტორინგის და კონტროლის შესაძლებლობა ეხმარება დროის დაზოგვაში. ბლეიდებით აგებულ სერვერულ ინფრასტუქტურაზე გადასვლა საშუალებას იძლევა მოხდეს სისტემის ინტეგრირებული მართვა და შევეშვათ $Intel$-სერვერების მუშაობის ძველ სქემას, როდესაც თითოეულ აპლიკაციაზე გამოყოფილია ცალკეული მანქანა. პრაქტიკაში ეს ნიშნავს: სერვერული რესურსების უფრო რაციონალურ გამოყენებას, რუტინული პროცედურების

(ისეთების, როგორიცაა სადენების შეერთება) რიცხვის შემცირებას, რომელიც უნდა შეასრულოს სისტემურმა ადმინისტრატორმა, მისი სამუშაო დროის ეკონომიას.

სისტემების და მონაცემთა შენახვის ქსელების გამოჩენა

გამოთვლითი სისტემის განვითარების თანამედროვე ისტორიის სხვა თავისებურებად, ბლეიდ-სერვერების გამოჩენესთან ერთად, გახდა სპეციალიზირებული სისტემების და მონაცემთა შენახვის ქსელების გამოჩენა. სერვერების შენახვის შიდა ქვესისტემები უკვე ვეღარ უზრუნველყოფდნენ სკალაციის საჭირო დონეს და მწარმოებლობას, დასამუშავებელი ინფორმაციის მზარდი მოცულობის პირობებში. ამის შედეგად გაჩნდა მონაცემთა შენახვის გარე სისტემები, რომლებიც ორიენტირებული იყვნენ მხოლოდ მონაცემთა შენახვის ამოცანების გადაწყვეტაზე და ინტერფიესის შეღწევადობის წამოდგენაზე მონაცემებზე მათი გამოყენების მიზნით.

მონაცემთა შენახვის სისტემა (მშს) - ეს არის ინფორმაციული რესურსების საიმედო შენახვის ორგანიზების პროგრამულ-აპარატული გადაწყვეტა და მათზე გარანტირებული წვდომის წარმოდგენა.

მონაცემთა შენახვის სისტემები წარმოადგენენ ცალკეულ კვანძებში შენახვის, გამოყოფის საიმედო მოწყობილობებს. მონაცემთა შენახვის სისტემა შეიძლება მივუერთოთ სერვერებს მრავალი ხერხით. ყველაზე ეფექტურია მიერთება ოპტიკური არხებით ($Fiber\ Channel$), რაც საშუალებას იძლევა მონაცემთა შენახვის სისტემებთან წვდომა გვქონდეს 4-8 გბიტ/წმ. სიჩქარით. მონაცემთა შენახვის სისტემებს ასევე აქვთ ძირითადი აპარატული კომპონენტების რეზერვირება - კვების რამდენიმე ბლოკი, $raid$ კონტროლერები,FC ადაპტერები და ოპტიკური პატჩკორდი (შემაერთებელი კაბელი) FC კომუტატორებთან მისაერთებლად.

ნახ.7. საწყისი დონის მონაცემთა შენახვის ტიპიური სისტემა ($Sun\ StorageTek$ 6140).

აღვნიშნოთ *მშს*- ის გამოყენების ძირითადი უპირატესობები:

მაღალი საიმედოობა და გაუმართაობისადმი ტოლერანტობა - ამის რეალიზება ხდება სისტემის ყველა კომპონენტის სრული ან ნაწილობრივი რეზერვაციით (კვების ბლოკების, შეღწევადობის გზების, პროცესორული მოდულების, დისკების, ქეშების და ა.შ.), ასევე მონიტორინგის მძლავრი სისტემით და შესაძლო და არსებული პრობლემების შეტყობინებით;

მონაცმთა **მაღალი ხელმისაწვდომობა** - მიიღწევა მონაცემთა მთლიანობის შენახვის გააზრებული ფუნქციებით (*RAID* ტექნოლოგიების გამოყენებით, დისკის საცავის შიგნით მონაცემების სრული და მყისიერი ასლების შექმნით, დისტანციური შენახვის სისტემასთან მონაცემების რეპლიკაციით და ა.შ.) და აპრატურის და პროგრამული უზრუნველყოფის შესაძლო დამატების (გაახლების) შესაძლებლობით მონაცემთა შენახვის სისტემის უწყვეტი მუშაობისას კომპლექსის გაჩერების გარეშე;

მართვის და კონტროლის მძლავრი საშუალებები - სისტემის მართვა *web*-ინტერფეისის ან ბრძანების ხაზის მეშვეობით, გაუმართაობის შესახებ ადმინისტრატორის შეტყობინების რამდენიმე ვარიანტის არჩევა, სრული მონიტორინგის სისტემა, რომელიც მუშაობს "რკინის" ტექნოლოგიის მუშაობის დიაგნოსტიკის დონეზე;

მაღალი პროდუქტიულობა - განისაზღვრება მყარი დისკების რაოდენობით, ქეშ-მეხსიერების მოცულობით, პროცესორული ქვესისტემის გამოთვლითი სიმძლავრით, შიდა (მყარი დისკებისათვის) და გარეშე (ჰოსტების მიერთებისათვის) ინტერფეისების რიცხვით, ასევე მოქნილი კონფიგურების შესაძლებლობით და მაქსიმალური პროდუქტიულობისათვის სისტემის კონფიგურაციით;

უპრობლემო სკალაცია - ჩვეულებრივ არსებობს: მყარი დისკების რიცხვის გაზრდის, ქეშ-მეხსიერების მოცულობის გაზრდის, მონაცემთა შენახვის არსებული სისტემის აპარატული მოდერნიზაციის, ჩასადებში მომუშავე სპეციალური პროგრამული უზრუნველყოფის მეშვეობით ზედმეტი გადაკონფიგურირების ან მონაცემთა შენახვის სისტემის რაიმე ფუნქციონალურობის დაკარგვის გარეშე ფუნქციონალის გაზრდის შესაძლებლობა. ეს მომენტი საშუალებას იძლევა საგრძნობლად გავწიოთ ეკონომია და უფრო მოქნილად დავაპროექტოთ მონცემთა შენახვის საკუთარი ქსელი.

ამჟამად, მონაცემთა შენახვის სისტემა წარმოადგენს ერთ-ერთ საკვნძო ელემენტს, რომელზედაც დამოკიდებულია კომპანიის ბიზნეს-პროცესის უწყვეტობა. თანამედროვე კორპორატიულ *IT* - ინფრასტუქტურაში მონაცემთა შენახვის სისტემები, როგორც წესი,

განცალკევებულია ძირითადი გამომთვლელი სერვერებისაგან, ადაპტირებული და მომართულია სხვადასხვა სპეციალიზირებული ამოცანებისათვის. მონაცემთა შენახვის სისტემები ახდენენ მრავალი ფუნქციის რეალიზებას, ისინი მნიშვნელოვან როლს თამაშობენ მონაცემთა ოპერატიულ სარეზერვო კოპირების და აღდგენის სისტემის, გაუმართაობისადმი ტოლერანტობის კლასტერების, მაღალი წდომის ვირტუალიზაციის ფერმების აგებაში.

თემა 4.მონაცემთა შენახვის ქსელები

SAN-ს გადაანაცვლებს შენახვის რესურსებს ჩვეულებრივი მომხმარებელების ქსელიდან და ახდენს მათ რეორგანიზებას დამოუკიდებელ, მაღალმწარმოებლურ ქსელად. ეს საშუალებას აძლევს თითოეულ სერვერს გააზიაროს საცავი ისე თითქოს ის იყოს უშუალოდ სერვერზე მიერთებული დისკი. როდესაც ჰოსტს სურს SAN-ის საცავ მოწყობილობაში შეღწევა იგი აგზავნის საცავ მოწყობილობაში ბლოკზე დაფუძნებულ შეღწევის ნებართვას.

საცავის ქსელის ზონა არის ტიპურად ასამბლერებული სამი ძირითადი კომპონენტის გამოყენებით: კაბელები, ძირითადი კომპიუტერის სალტის გადამყვანები (HBAs) და კომუტატორები მიერთებული საცავის მასივებსა და სერვერებზე. თითოეული გადართვის და საცავი სისტემა SAN-ზე უნდა იყოს ურთერთდაკავშირებული და ფიზიკურმა ურთიერთკავშირმა უნდა უზრუნველყოს გამტარიანობის დონე, რომელიც შესაბამისად გაუმკლავდება მონაცემების დამუშვების პიკურ ოპერაციებს. IT ადმინისტრატორი მართავს საცავი ზონის ქსელებს ცენტრალურად.

თავდაპირველად შენახვის მასივები იყო სისტემები მყარი დისკით, მაგრამ სულ უფრო ხშირად ივსებოდა მყარი მდგომარეობის დამგროვებლებით (SSDs).

რისთვის გამოიყენება მონაცემთა შენახვის ქსელები

ქსელ Fiber Channel-ს (FC) აქვს ძვირადღირებულის, კომპლექსურის და რთულად სამართვის რეპუტაცია. ეზერნეტზე დაფუძნებულმა iSCSI შეამცირა ეს გამოწვევები SCSI ბრძანებების IP პაკეტებში ჩასმით, რაც არ მოითხოვს FC კავშირს.

iSCSI-ის წარმოქმნა ნიშნავს, რომ ორი ქსელის -Ethernet ლოკალური ქსელი მომხმარებელთა კომუნიკაციისათვის და FC SAN შენახვისთვის- შესწავალა, აგება და მართვის ნაცვლად ორგანიზაციას შეუძლია გამოიყენოს არსებული ცოდნა და ინფრასრუქტურა როგორც LAN-ის ასევე SAN-ისათვის.

ორგანიზაციები იყენებენ SAN-ს განაწილებული აპლიკაციებისათვის, რომლებსაც სჭირდებათ ლოკალური ქსელის სწრაფი შესრულება . SAN-ის აუმჯობესებს აპლიკაციების ხელმისაწვდომობას მონაცემთა ორი არხის მეშვეობით. მათ აგრეთვე შეუძლიათ

გააუმჯობესონ აპლიკაციების მუშაობა, რადგან ისინი საშუალებას აძლევს IT ადმინისტრატორს განტვირთონ შენახვის ფუნქცია და დაყონ ქსელი.

გარდა ამისა, SAN ეხმარება საცავის ეფექტურობის ზრდას და გამოყენებას, ვინაიდან იგი საშუალებას აძლევს ადმინისტრატორს თავი მოუყაროს რესურსებს და შექმნას მრავალშრიანი საცავი. SAN ასევე აუმჯობესებს მონაცემთა დაცვას და უსაფრთხოებას. საბოლოოდ, SAN-მა შეიძლება მოიცვას რამოდენიმე საიტი, რომელიც ეხმარება კომპანიებს მათი ბიზნესის უწყვეტი განვითარების სტრატეგიის შემუშავებაში.

SAN კომუტატორების არსი

SAN კომუტატორები აერთებენ სერვერებს და გაზიარებული საცავი მოწყობილობების ერთობლიობას (Pools). SAN კომუტატორების ერთადერთი დანიშნულებაა საცავის ტრაფიკის გადაადგილება. SAN კომუტატორები ხშირად არიან Fiber Channel კომუტატორები, რომლებიც თავსებადია FC პროტოკოლთან, რასაც ეფუძნება მრავალი SAN ქსელი. კომუტატორი ამოწმებს მონაცემთა პაკეტს და ამოიცნობს მათ წარმოშობას და დანიშნულებას. მაშინ, კომუტატორი მიმართავს პაკეტებს საჭირო საცავ სისტემაში. FC კომუტატორების განკუთვნილია მაღალი გამტარიანობის ქსელებისათვის.

SAN კომუტატორები შეიძლება იყოს აგრეთვე Ethernet-ზე დაფუძნებული. ასეთმა კომუტატორებმა უნდა გადაამუშონ მხოლოდ მოძრაობა IP SAN-ზე რათა შეინარჩუნონ პროგნოზირებადი მწარმოებლურობა. Ethernet კომუტატორები მიაწოდებს მოძრაობას IP მისამართებზე; ისინი აღიქვავენ iSCSI საცავის მიზანს IP მისამართად.

ორგანიზაცებს შეუძლიათ გააერთიანონ მრავალი კომუტატორი რათა შექმნან SAN-ის დიდი ქსელი, რომელიც უკავშირდება მრავალ სერვერს და საცავ პორტს.

ვირტუალური SAN

ვირტუალური საცავების ქსელი (VASAN) არის პროგრამულად განსაზღვრული საცავი, რომელიც მოთავსებულია ჰიპერვიზორის ზევით, როგორიცაა VMware ESXi ან Microsoft Hyper-V.

უმეტეს შემთხვევებში VSAN არ არის დამოკიდებული მყარ მოწყობილობებზე. სანამ ჰიპერვიზორი აღიარებს და მხარს უჭერს საცავ მოწყობილობას, VSAN-ს შეუძლია მისი გამოყენება, თუმცა ყოველ ვენდორს აქვს საკუთარი მოთხოვნები.

ერთიანი SAN

ერთიანი SAN ეფუძნება ერთიანი საცავის კონცეფციას, რომელიც წარმოადგენს ფაილების საცავს და ბლოკურ საცავს ერთ საცავ მოწყობილობაში.

უნიფიცირებული SAN ამ კონცეფციას ერთი ნაბიჯით წინ სწევს არა მხოლოდ ცალკეული ლოგიკური ერთეულის ნომრის გამოყოფით, მაგალითად სხვა ნებისმიერი SAN, არამედ ფილის სისტემაზე დაფუძნებით, NAS-ის მსგავსი საცავი.

კონვერგენტული SAN

საცავი ქსელები ჩვეულებრივ გამოყოფილია ეზერნეტის ქსელებისაგან. კონვერგენტული (თანხვედრილი) SAN ქსელისათვის გამოიყენებს ჩვეულებრივ ქსელურ ინფრასტრუქტურას და SAN ტრაფიკს რათა აღმოფხვრას გადაჭარბებული ინფრასტრუქტურა და შეამციროს ღირებულება და სირთულე.

SAN ხშირად იყენებს Fiber Channel-ს, სადაც მონაცემთა ქსელები ყოველთვის ეფუძნება Ethernet-ს. თანხვედრილი SAN იყენებს Fiber Channel Ethernet-ის მეშვეობით, რომელიც ახდენს FC-ს ზედმეტ დატვირთვის ინკაფსულაციას Ethernet-ის ჩარჩოებში. თანხვედრილი SAN თითქმის ყოველთვის ეფუძნება 10 გიგაბიტიან Ethernet-ს და მრავალი ქსელის პორტი ერთაიანდება გამტარუნარიანობის გასაზრდელად.

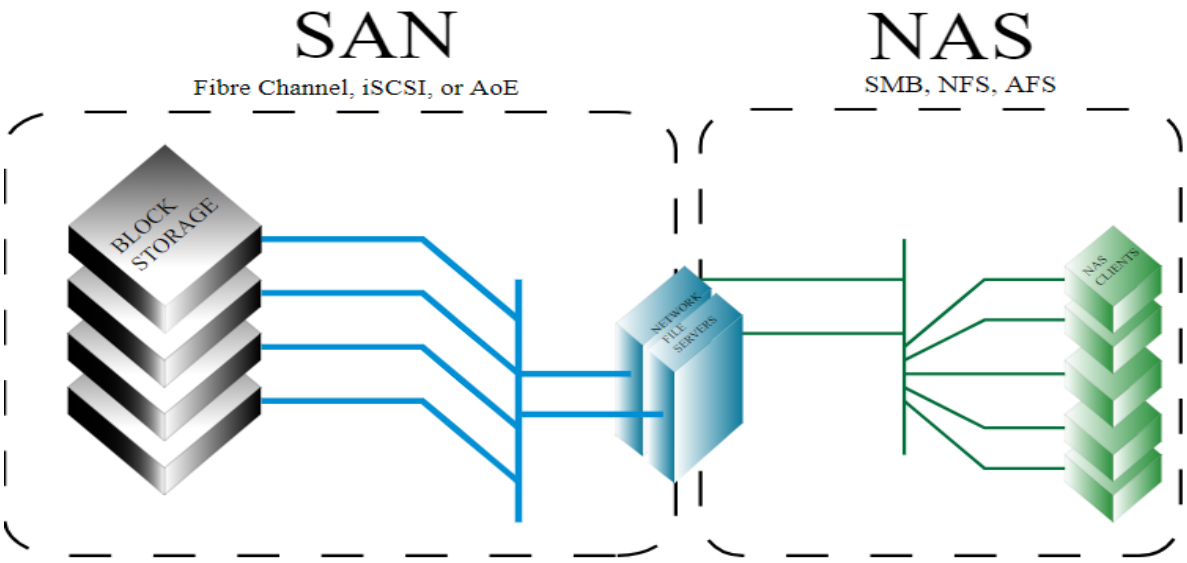
SAN-ის პლიუსები და მინუსები

SAN-ის გამოყენების მთავარი სარგებელია საწყისი raw ტევადობის განიხილვა როგორც რესურსების აუზის, რომლის მართვაც და განაწილებაც IT-ს შეუძლია საჭიროებისამებრ. SAN ასევე მაღალ მასშტაბირებულია, რადგან საჭიროების შემთხვევაში შეიძლება სიმძლავრის დამატება.

SAN-ის მათავარი უარყოფითი მხარე არის ღირებულება და სირთულე. SAN-ის მოწყობილობები საკმაოდ ძვირია და SAN-ის აწყობა და მართვა მოითხოვს სპეციალიზებულ უნარების ნაკრებს.

SAN NAS-ის წინააღმდეგ

ტერმინები SAN და NAS ზოგჯერ ერევათ ვინაიდან მათი შემოკლებული ჩანაწერი ჰგავს ერთმანეთს. NAS შედგება შენახვის მოწყობილობისგან, რომელიც პირდაპირ ქსელის კომუტატორშია ჩართულია. თუმცა არსებობს გამონაკლისები NAS მოწყობილობები ხშირად გამოიყენება როგორც ფაილური სერვერები, მაშინ როცა SAN გამოიყენება სტრუქტურული მონაცემების შესანახად მონაცემთა ბაზაში.



ნახ.8. SAN და NAS შედარება.

SAN უფრო მეტად რთული და ძვირადღირებულია ვიდრე NAS. SAN შედგება: გამოყოფილი კაბელისაგან - ჩვეულებრივ Fiber Channel-ისაგან, მაგრამ ethrtnet-ი შეიძლება გამოყენებულ იქნეს iSCSI ან FCoE SAN-ში; გამოყოფილი კომუტატორებისა და საცავი მოწყობილობისაგან. SANs ძალიან სკალირებადია და საშუალებას გვაძლევს შევინახოთ საცავები როგორც LUN- ები.

პირიქით, NAS საცავი ჩვეულებრივ წარმოადგენილია როგორც ფაილური სისტემა, თუმცა NAS-ის ზოგიერთი მოწყობილობა მხარსუჭერს ბლოკ საცავს.

მთავარი გამყიდველები და პროდუქტები

SAN მომწოდებლების ბაზარზე მოხდა მრავალი გაერთიანება და დღეს დომინირებს რამოდენიმე მსხვილი IT კომპანია.

2016 წლის სექტემბერში დელმა დაასრულა 60 მილიარდი აშშ დოლარის ღირებულების EMC-ს შეძენა, რომელიც იყო ყველაზე დიდი IT შენაძენი. EMC იყო ლიდერი საცავების მასივების ბაზარზე და Dell EMC ახლა გამოიმუშვებს უდიდეს შემოსავალს და აქვს უდიდესი საცავი სისტემის პორტფელი. Dell EMC საცავში ახლა შედის VMAX, Unity და XtremlO SAN მასივები ძველი EMC და Compellent-ს Dell-სგან.

Dell-ის სერვერის კონკურენტმა, Hewlett Packard Enterprise-მაც (HPE), შეავსო საკუთარი SAN პროდუქტების პორტფელი შენაძენებით. HPE-მ შეიძნა Nimbel საცავი 2017 1,2 მლრდ. დოლარად. ეს მოჰყვა 2010-ში 3PAR-ის შეძენას 2.35 მლრდ.დოლარად. 3PAR და Nimbel პროდუქტების ხაზი არის ამჟამად HPE-ს მთავარი მასივის პლატფორები.

NetApp წარმოადგენს მონაცემების შენახვის სისტემის მსხვილ მომწოდებელს, რომელიც დარჩა Dell-ის EMC-თან შერწყმის შემდეგ. NetApp-მა დაიწყო საქმიანობა როგორც ქსელის ინსტრუმენტმა 1992 წლიდან, და იგი შედიოდა NAS-ის პირველ მომწოდებლებში. 2002 წელს მან დაამატა ბლოკური შენახვის შესაძლებლობა თავის პლატფორმა FAS-ს (Fabric-Attached Storage) და FAS მასივები ხელმისაწვდომია FC, iSCSI ან NAS კავშირით, ან ამ სამის ნებისმიერი კომბინაციით.

IBM და Hitachi Vantara (მანამდეHitachi Data Systems) წარმოადგენენ სხვა მსხვილ SAN მასივების მომწოდებლებს. IBM and Hitachi ძირითადად ფოკუსირებულები არინ მსხვილ საწარმოებზე და მეინფრეიმზე მიერთებული საცავებზე, თუმცა აფრათოებს საკუთარ პორტფელს ღია სისტემისათვის ფლეშ მასივებით.

SAN (Storage area network) არის მონაცემთა შენახვის მოწყობილობების ქსელი. SAN საცავი შეიძლება შედგებოდეს რამოდენიმე სერვერისაგან, რომელიც დაკავშირებულია შენახვის მოწყობილობების ცენტრალურ აუზთან, შესაძლოა მოიცავდეს ათასობით სერვერს მიმართულს TB ან მეტ საცავთან.

SAN-ში მონაცემები წარმოდგენილია საცავი მოწყობილობიდან ჰოსტისაკენ მოცემულია ისე რომ საცავი გამოიყურება როგორც ადგლობრივად მიერთებული. ეს მიიღწევა მონაცემების სხვადასხვა სახის ვირტუალიზაციიდან. ამგვარად, SAN საცავი არის მაღალსიჩქარიანი ქსელი,

რომელიც უზრუნველყოფს ქსელის წვდომას საცავზე. ზოგიერთ შემთხვევაში SAN შეიძლება იყოს იმდენად დიდი რომ მოიცვას მრავალი ვებგვერდი, ასევე შიდა მონაცემთა ცენტრები და ღრუბელი.

მონაცემთა შენახვის ქსელები განსხვავდებიან პირდაპირი საცავისაგან (DAS- Direct Attached Storage). DAS-ში მონაცემები პირდაპირ არის მიერთებული ერთ სერვერთან. მეორე მხრივ, SAN წარმოადგენს საცავ მოწყობილობებს ისე რომ საცავი ხდება ადგილობრივად მიერთებული. საცავის ჰოსტთან ეს გამარტივებული პერსონალიზაცია მიიღწევა ვირტუალიზაციის სხვადასხვა მეთოდით.

SAN ასევე განსხვავდება ქსელზე დაკავშირებული საცავისაგან (NAS). მაშინ როცა NAS ასევე გააქვს საცავი მოწყობილობი სერვერიდან მონაცემების ცენტრალური აუზის შესაქმნელად, NAS საცავი პირდაპირ უკავშირდება ქსელს (LAN). SAN საცავში, ტევადობა გაერთიანებულია და უზრუნველყოფილია გამოყოფილი ქსელით. ეს იძლევა სწრაფი კომუნიკაციის საშუალებას სწრაფი მედიის მეშვეობით.

SAN საცავის უპირატესობა

SAN მრავალრიცხოვანია, აქედანაა მისი პოპულარობა საწარმოებში, SAN-ის სარგებელი მოიცავს:

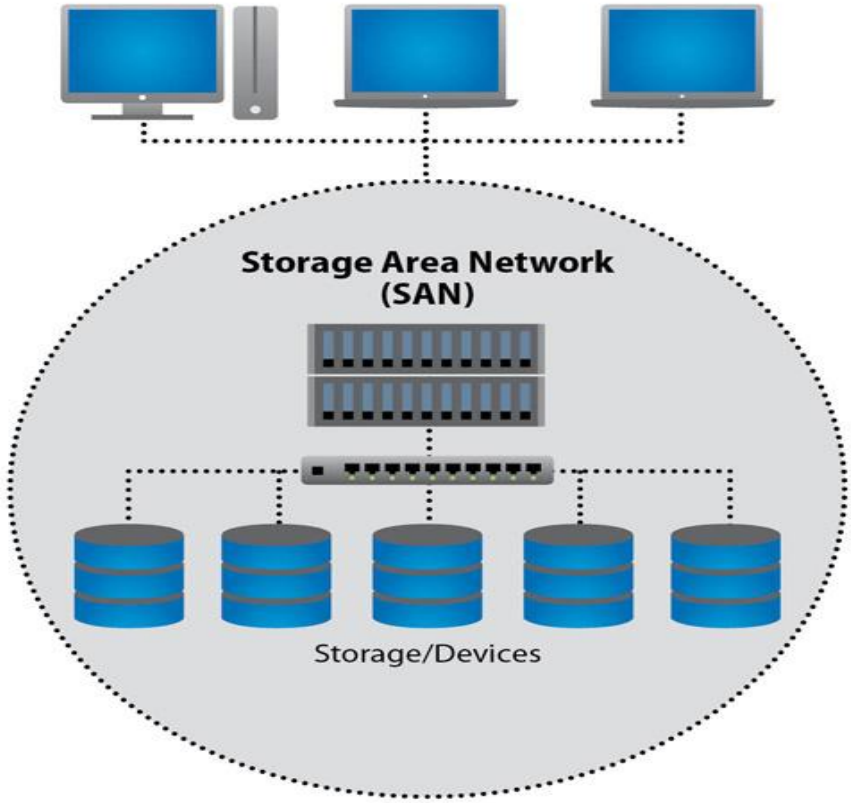
- გამტარის ზოლში შევიწროვებული ადგილების აღმოფხვრას, რომელიც დაკავშირებულია LAN- ბაზაზე საცავის სერვერთან.
- CSCI სალტე-ბაზაზე შესრულების მიერ არ არის დაწესებული მასშტაბირების ლიმიტი.
- მაღალ ხელმისაწვდომობას
- უფრო მეტი გაუმართაობისადმი ტოლერანტობას
- საცავისის ცენტრალიზებულ მართვას
- უფრო სწრაძ სარეზერვო კოპირებას
- გლობალურ ფაილ სისტემას
- მონაცემთა სწრაფ მიგრაციას
- გაუმართაობისადმი ტოლერანტობას
- მონაცემთა უკეთეს დაცვას
- საცავის გაუმჯობესებულ გამოყენებას
- უფრო მეტ მასშტაბირებას

- გადიდებულ ხემისაწვდომობას აპლიკაციებზე, როგორიცაა მონაცემთა ორმაგი გასავალი
- აპლიკაციის გაზრდილ მწარმოებლურობას შენახვის ფუნქციის განტვირთვის ან ქსელის გაყოფის ხარჯზე
- უკეთესი მონაცემთა დაცვა და ავარიული აღდგენა (DR)

SAN-ის უპირატესობისათვის განვიხილოთ საცავის მიგრაცია. თუ მონაცემი ინახება მრავალ სერვერზე, ადმინისტრატორისათვის საკმაოდ შრომატევადია მათი აღება თითოეული სერვერიდან და მათი გადატანა ახალ სახლში. ეს ალბათ მოითხოვს ისეთ ნაბიჯს, როგორიცაა დაშლა ფაილური სისტემა საცავის გამოყენებით, მოწყობილობის გამორთვა, მისი გადაადგილება, შემდეგ მისი დაკავშირება სხვა ჰოსტთან და ფალური სისტემის გაშვება ახალ კომპიუტერზე. SAN საცავის გადაწყვეტაში, მარტივია მთლიანი დიდი მასივის გადატანა ერთი ჰოსტიდან სხვაზე. რისი გაკეთებაც მოგიწევთ არის ფაილური სისტემის დაშლა, SAN-ის სწრაფი რეკონფიგურირება, შემდეგ მონაცემების გადატანა ახალ ჰოსტზე. ეს ზოგავს საცავის ადმნისტრატორის დიდინოდენობით დროს.

ეს არქიტექტურა ხდება უფრო და უფრო მიშვნელოვანი რამდენადაც საცავის ზომა იზრდება. მოუხერხებელია მონაცემთა მრავალრიცხოვანი TB-ს მართვა სერვერზე სერვერის ბაზაზე. მას სჭირდება შენახვის ქსელი, რათა მან შეასრულოს მძიმე გადატანა და ააცილოს მძიმე შრომა. მაგალითად, თუ გჭირდებათ სხვა მასივიდან საცავის დამატება სერვერზე SAN არქიტექტურა საშუალებას გაძლევთ განათავსოთ მოწყობილობის ლოგიკური რიცხვი (LUN) მრავალი მასივიდან ამ ერთ სერვერზე.

რა არის LUN? LUN არის უნიკალური ინდიფიკატორი, რომელიც აღნიშნავს შენახვის ცალკეულ მოწყობილობას ან შენახვის ფიზიკური ან ვირტუალური ელემენტების ნაკრებს, რომლებიც ასრულებენ შეტანა/გამოტანის ბრძანებებს ჰოსტ კომპიუტერიდან. ლოგიკური ერთეული, რომელიც იდენტიფიცირებულია შეიძლება იყოს დამგროვებელ მოწყობილობაზე ტევადობის ბლოკი, მთლიანი დამგროვებელი მოწყობილობა ან რამოდენიმე მყარი დისკის ნაწილი, მყარსხეულიანი დამგროვებელი ან ლენტური საცავი, განთავსებული ერთ ან რამოდენიმე საცავ სისტემაში.



ნახ.9. SAN.

მონაცემთა შეანხვის ქსელი აერთიანებს შეანხვის მრავალ მოწყობილობას და უზრუნველყოფს უფრო სწრაფ კავშირს.

ფაილი ბლოკის წინააღმდეგ

არსებობს სხვა საკვანძო განსხვავება სერვერზე (ან NAS ყუთი) მონაცემთა შენახვასა და SAN-ში მონაცემების შენახვას შორის. პირველი იყენებს ფაილის დონის საცავს და უკანასკნელი - ბლოკის დონის საცავს.

- ფაილის დონის საცავი მოთავსებულია მყარ მოწობილობაში და NAS სისტემებში, საცავი დისკი კონფიგურირებულია ისეთ პროტოკოლთან როგირიცაა NFS ან CIFS იმგვარად, რომ ფაილები შენახული და ხელმისაწვდომია ერთ მასაში (in bulk), მაგალითად, ფაილზე ფაილის ბაზაზე. ეს მიდგომა მარტივი და ადვილი გამოსაყენებელია.
- ბლოკის დონის საცავი ქმნის საწყისი მეხსიერების მოცულობას, სადაც მონაცემის თითოეული ბლოკი კონტროლირდება ოპერატიული სისტემის მიერ თითქოს ეს იყოს

გამოცალკევებული მყარი დისკი. მონაცემთა ეს ბლოკები არ არის მიბმული რომელიმე კონკრეტულ ფაილზე. ბლოკური საცავი შემდეგ მართავს LUN-ებს, განსხვავებით ინდივიდუალური ფაილებისა, რომლებიც მართულია NAS სისტემებში.

ბლოკის დონეზე საცავის სარგებელი მოიცავს:

- მოქნილი მენეჯმენტი
- მონაცემთა ბაზების იოლი შენახვის მენეჯმენტი
- მონაცემების სწრაფი და უფრო საიმედო გადაცემა
- თითოეული საცავის სიდიდის დამოუკიდებელ დისკად დაყოფის შესაძლებლობა, რომელიც კონტროლირდება გარე სერვერის ოპერაციული სისტემით.
- მარტივია წვდომის და კონტროლირების პრივილეგიების მართვა.

იგი შეიძლება იყოს ან/და შეთავაზება- ბლოკური საცავი ან ფაილური საცავი. მაგრამ ბოლო დროს შემუშავდა საცავსისტემები, რომელთაც შეუძლიათ მუშობა ფაილურ საცავებთან და ბლოკურ დონეზეც ერთ მოწყობილობაში. ეს უნიფიცირებული საცავი სისტემები და ჰიდროკონვერგირებული სისტემები უფრო და უფრო გავრცელებული ხდება.

iSCSI SAN

ისევე როგორც FC SAN არსებობს ასევე SAN ინტერნეტის მცირე გამოთვლითი სისტემების ინტერფეისით (iSCCI- Internet Small Computing System Interface, ასევე ზოგჯერ ცნობილია როგორც IP SAN). საცავი iSCSI საშუალებას იძლევა მონაცემები გადაეცეს დამმახსოვრებელ მოწყობილობას და დამგროვებელი მოწყობილობიდან IP ქსელს ტრაფიკის სერიალიზაციით (serializing) შეერთება SCSI-დან. ბოლო ათწლეულის განმავლობაში ჩვეულებრივ, მცირე და საშუალო ზომის ბიზნესში, როგორც FCC-ს უფრო იაფი ალტერნატივა, iSCSI SAN-ის პოპულარობა გაიზარდა. ეს გამოიწვია იმან რომ FC SAN-ს გააჩნია მართვის სირთულის რეპუტაცია, რომელსაც სჭირდება მაღალკვალიფიციური (კარგად ანაზღაურებადი) სპეციალისტები და ამასთან არის საკმაოდ ძვირი. Ethernet-ის გამოყენებით iSCSI SAN შეუძლია გადასცეს SCSI ბრძანებები IP პაკეტებში ისე რომ აღარ არის საჭირო რაიმე FC კავშირი.

- უპირატეოსობა: არ არის FC ქსელის შესწავლის, შექმნის და მართვის საჭიროება; ერთი და იმავე სადენის გამოყენება, როგორც Ethernet- ზე დაფუძნებული LAN-ისათვის და ასევე საცავისათვის.
- ნაკლი: LAN-ის დაბინძურება საკმაოდ დიდი ტრაფიკით - თუმცა iSCSI საცავი უფრო მეტად გამოიყენება პატარა და საშუალო ზომის ორგანიზაციების მიერ.

მეორე მხრივ FC SAN უფრო ხშირად გამოიყენება მსხვილი ორგანიზაციების მიერ, ისევე როგორც მათ მიერ ვისაც აქვს განაწილებული აპლიკაციები რომელთაც სჭირდება სწრაფი ადგილობრივი ქსელი. მონაცემების მრავალი გამტარის შეთავაზებით FC SAN სთავაზობს აპლიკაციების უკეთესს მუშაობას და IP ქსელიდან საცავის ფუნქციის გათიშვას.

მიუხედავად იმისა ორგანიზაცია იყენებს თუ არა iSCSCI ან FC SAN-ს საცავის ეფექტურობა და გამოყენება იზრდება რამდენადაც ადმინისტრატორს შეუძლია გააერთიანოს რესურსები და დააყენოს მრავალდონიანი საცავი:

- სუპერ სწრაფი საცავის ზედა დონე განკუთვნილია მონაცემების მცირე ნაკრებისათვის და კრიტიკულად მნიშვნელოვანი აპლიკაციებისათვის.
- შემდეგ მას მოჰყვება მიმდევრობითი დონეები შედარებით ნელი შენახვისა და მაღალი ტევადობით.

SAN პროგრამა, რა თქმა უნდა, ასევე საჭიროა სერვერების, საცავი მოწყობილობების მოსაწესრიგებლად ისეთი ფუქციისათვის, როგორიცაა მონაცემების გადაცემა. LUN-ის გადასაადგილებლად და არსებული ფაილური სისტემაზე ტევადობის დასამატებლად, საჭიროა მოცულობის მართვის პროგრამა. RAID ასევე გამოიყენება SAN-ში, როგორიცაა პროგრამული დონის RAID, რათა უზრუნველყოს RAID 0 ზოლი.[4]

SAN - ეს არის მაღალსიჩქარიანი კომუტირებადი მონაცემთა გადაცემის ქსელი, რომელიც აერთიანებს სერვერებს, სამუშაო სადგურებს, დისკურ საცავებს და ფირის ბიბლიოთეკებს. მონაცემთა გაცვლა მიმდინარეობს *Fibre Channel* პროტოკოლის მიხედვით, რომელიც ოპტიმიზირებულია შეტყობინებების სწრაფი გარანტირებული გადაცემისათვის და საშუალებას იძლევა გადავცეთ ინფორმაცია რამდენიმე მეტრიდან ასობით კილომეტრის მანძილზე.

[4] http://www.enterprisestorageforum.com/storage-networking/storage-area-networks-in-the-enterprise.html

მონაცემთა შენახვის ქსელის განვითარების მამოძრავებელ ძალად იქცა ისეთი საქმიანი ინფორმაციის სწრაფი ზრდა (როგორიცაა ელექტრონული ფოსტა, მონაცემთა ბაზა და ზედატვირთული ფაილური სერვერები), რომლებიც მოითხოვენ ბლოკურ დონეზე მაღლსიჩქარიან წვდომას დისკურ მოწყობილობებისადმი. წინათ საწარმოებში ჩნდებოდნენ მაღალპროდუქტიული დისკური მასივების „კუნძულები“ $SCSI$. თითოეული ასეთი მასივი გამოყოფილი იყო კონკრეტული აპლიკაციისათვის მისთვის ჩანდა, როგორც „ვირტუალური მყარი დისკების“ გარკვეული რაოდენობა. მონაცემთა შენახვის ქსელი ($Storage\ Area\ Network$ ანუ SAN) ასეთი „კუნძულების გაერთიანების საშუალებას იძლევა მაღლსიჩქარიანი ქსელის საშუალებებით. SAN-ის საფუძველს წარმოადგენს მოწყობილობების ბოჭკოვან-ოპტიკური შეერთებები Fibre Chanel ინტერფეისით, რომელიც უზრუნველყოფს ობიექტებს შორის ინფორმაციის გადაცემას 1, 2, 4, ან 8 Gbit/sec. სიჩქარით. შენახვის ქსელები საშუალებას იძლევიან მაღლა ავწიოთ შენახვის სისტემის რესურსების გამოყენების ეფექტურობა, რამდენედაც ქსელის ნებისმიერ კვანძზე ნებისმიერი რესურსის გამოყოფის შესაძლებლობას იძლევიან.

განვიხილოთ SAN-ის ძირითადი უპირატესობანი:

• **პროდუქტიულობა**. SAN-ის ტექნოლოგიები საშუალებას იძლევა უზრუნველყოთ მაღალი პროდუქტიულობა მონაცემთა შენახვისა და გადაგზავნისათვის ამოცანების გადასაწვეტად.

• **სკალაცია.** მონაცემთა შენახვის ქსელი უზრუნველყოფს შენახვის ქვესისტემის გაფართოების მოხერხებულობას, საშუალებას იძლევა ადვილად გამოვიყენოთ ადრე შეძენილი მოწყობილობები ახალი შენახვის მოწყობილობებთან ერთად.

• **მოქნილობა.** მონაცემთა შენახვის ქსელის ერთობლივი გამოყენება, როგორც წესი, ამარტივებს ადმინისტრირებას და მატებს მოქნილობას, ვინაიდან კაბელების და დისკის მასივების ფიზიკური ტრანსპორტირება და გადაკომუტირება ერთი სერვერიდან მეორეზე არაა საჭირო. SAN ახალი სერვერების და დისკის მასივების სისტემის გაუჩერებლად ქსელთან მიერთების საშუალებას იძლევა..

• **ცენტრალიზებული ჩატვირთვა.** სხვა უპირატესობას წარმოადგენს სერვერის ჩატვირთვა პირდაპირ შენახვის ქსელიდან. ასეთი კონფიგურაციის დროს ადვილადაა შესაძლებელი

გაუმართავი სერვერის შეცვლა, SAN-ის იმგვარად გადაკონფიგურირება, რომ სერვერ-შემცვლელი ჩაიტვირთვირთოს გაუმართავი სერვერის ლოგიკური დისკიდან.

• **გაუმართაობისადმი ტოლერანტობა.** შენახვის ქსელი გვეხმარება უფრო ეფექტურად აღვადგინოთ გაუმართაობის შემდეგ შრომისუნარიანობა. SAN-ს შეუძლია შეაღწიოს მოშორებულ უბანში შენახვის მეორადი მოწყობილობით. ამ შემთხვევაში შეიძლება გამოვიყენოთ რეპლიკაცია რომელიც რეალიზებულია მასივების კონტროლიორების დონეზე, ან სპეციალური აპარატული მოწყობილობების საშუალებით ვისარგებლოთ. ასეთ გადაწყვეტილებებზე მოთხოვნა საგრძნობლად გაიზარდა აშშ-ში 2001 წლის 11 სექტემბრის მოვლენების შემდეგ.

• **მართვა.** SAN-ის ტექნოლოგიები საშუალებას გვაძლევს უზრუნველყოთ მთელი მონაცემთა შენახვის ქვესისტემის ცენტრალიზებული მართვა.

SAN ტოპოლოგიები

კონცეპტუალურად, SAN არის მაღალ სიჩქარიანი ქსელი განკუთვნილია ინფორმაციის სამართავად. აპლიკაციაში SAN არის ტექნოლოგიის კომბინაცია, მოწყობილობის, პროგრამის და ქსელის კომპონენტების ჩათვლით, რომელიც უზრუნველყოფს სერვერისა და საცავის ელემენტების ნებისმიერ ურთიერთკავშირს.

ამიტომ SAN-ის ნებისმიერი რეალიზაცია შექმნილია სპეციალურად მისი გამომყენებელი კლიენიტის ბიზნეს-მოთხოვნის შესაბამისად, ამისათვის გამოიყენება კომპონენტების ის ნაკრები რომელიც შეირჩევა სხვადასხვა მომწოდებლების ან მწარმოებლებისაგან. SAN-ის დიზაინი დამოკიდებულია ბიზნესზე და კლიენტის ტექნიკურ მომთხოვნილებებზე.

SAN უზრუნველყობს მაღალი სიჩქარით მონაცემთა გადაცემებს სერვერებს და საცავ მოწყობილობებს შორის შემდეგი სამი გზით:

- **სერვერი საცავს:** ეს არის საცავ მოწყობილობებს შორის ინტერაქციის ტრადიციული მოდელი. უპირატესობა ის არის, რომ მრავალ სერვერს შეუძლია მიმართოს ერთიდაიგივე საცავ მოწყობილობას სერიულად ან ერთდროულად.
- **სერვერი სერვერს**: SAN შეიძლება გამოიყენებოდეს სერვერებს შორის მაღალ სიჩქარიანი, მაღალ გამტარიანი კომუნიკაციისათვის, უზრუნველყოფს

პრაქტიკულ ინფრასტრუქტურას ისეთი ტექნოლოგიების გათვალისწინებით, როგორიცაა სერვერული კლასტერული, სარკული ასახვა, დატვირთვის დაბალანსება.

- **საცავი საცავს**: SAN საშუალებას აძლევს მონაცემებს გადაადგილდნენ სერვერის ინტერვენციის გარეშე, ათავისუფლებს რა ამით სერვერის პროცესორის ციკლს სხვა აქტივობებისთავის, რგორიცაა აპლიკაციის დამუშავება. ამ კომუნიკაციის ზოგიერთი მაგალითი მოიცავს დისკის მოწყობილობას, რომელიც ახდენს მონაცემების რეზერვირებას ჩამწერ მოწყობილობაზე სერვერის ინტერვენციის გარეშე, ან დაშორებული მოწყობილობის mirroring-ს SAN-ის მეშვეობით.

განვიხილოთ მონაცემთა შენახვის ქსელების ზოგიერთი ტოპოლოგია. ერთკომუტატორიანი სტრუქტურა (single-switch fabric) შედგება ერთი Fibre Channel კომუტატორისაგან, სერვერისაგან და მონაცემთა შენახვის სისტემისაგან. ჩვეულებრივ ეს ტოპოლოგია წარმოადგენს ყველა სტანდარტული გადაწყვეტილებისათვის საბაზისოს - სხვა ტოპოლოგიები იქმნება ერთკომუტატორნიანი უჯრედების გაერთიანებით.

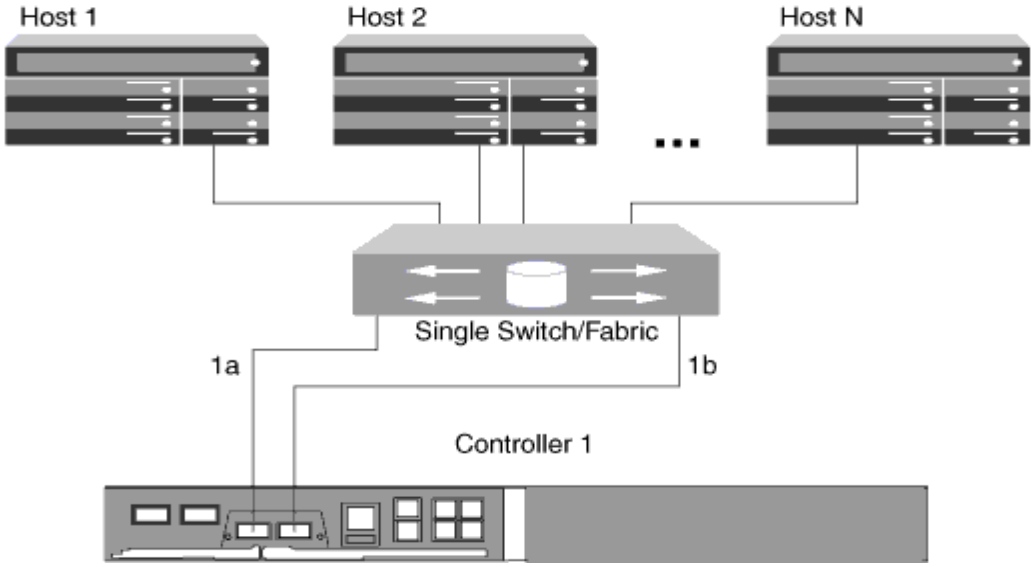
თქვენ შეგიძლიათ დააკონფიგუროთ FC და FC-NVMe SAN ჰოსტები ერთი კვანძის მეშვეობით ერთ ან მეტ სტრუქტურის გავლით. საჭიროა N-Port ID (NPIV)-ის ვირტუალიზაცა და უნდა ჩართოთ ყველა FC გადამრთველები სტრუქტურაში. თქვენ შეგიძლიათ პირდაპირ მიუერთოთ FC ან FC-NMVE SAN ჰოსტები ცალკეულ კვანძს FC გადამრთველის გმოყენების გარეშე.

ერთი-სტრუქტურა ერთი-კვანძი კონფიგურაცია

ერთი-სტრუქტურა ერთი-კვანძი კონფიგურაციაში ერთი გადამრთველი, რომელიც აკავშირებს ერთ კვანძს ერთ ან მეტ ჰოსტთან. ვინაიდან აქ არის მხოლოდ ერთი გადამრთველი, ეს კონფიგურაცია არის სრულად გადატვირთული. ყველა აპარატული პლატფორმა, რომელიც მხარს უჭერს FC და FC-NVMe-ს თავის მხრივ მხარს უჭერს ერთი-სტრუქტურა ერთი-კვანძი კონფიგურაციებს. თუმცა, FAS2240 პლატფორმა მოითხოვს X1150A-R6 გაფართოების ადაპტერს ერთი-სტრუქტურა ერთი-კვანძი კონფიგურაციის მხარდასაჭერად.

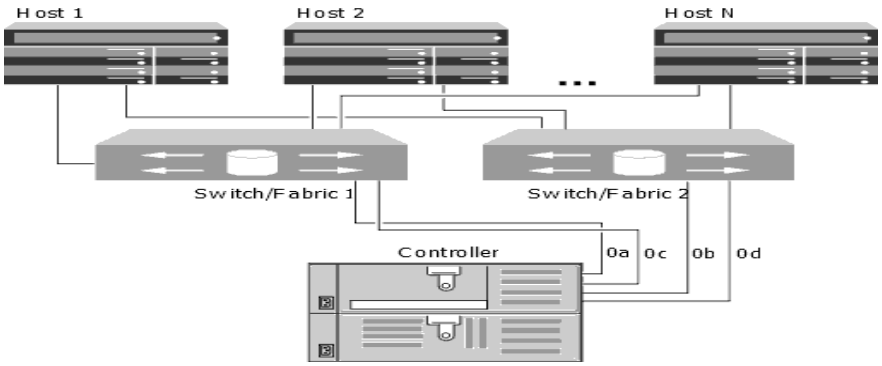
მოცემული ნახატი გვიჩვენებს FAS2240 ერთი-სტრუქტურის ერთი-კვანძი კონფიგურაციას. იგი გვიჩვენებს საცავის კონტროლერებს გვრდიგვერდ, სწორედ ისე როგორც

მონტაჟდებიან FAS2240-2-ში. FAS2240-4-ში კონტორლერები დამონტაჟებულია ერთმანეთზე. არ არის განსხვავება SAN კონფიგურაციაში ამ ორი მოდელისათვის.



ნახ.10. Multifabric ერთ-კვანძიანი კონფიგურაცია.

მრავალსტრუქტურიან ერთ-კვანძიან კონფიგურაციაში არის რამოდენიმე გადამრთველი, რომელიც აკავშირებს ერთ კვანძს ერთ ან რამდენიმე ჰოსტთან. სიმარტივისათვის, მოცემული ნახატი გვიჩვენებს მრავალსტრუქტურიან ერთ-კვანძიან კონფიგურაციას მხოლოდ ერთი სტრუქტურით, მაგრამ თქვენ შეგიძლიათ გქონდეთ ორი ან მეტი სტრუქტურა მრავალსტრუქტურიან კონფიგურაციაში. ამ ნახაზზე საცავის კონტროლერი განთავსებულია ზედა შასიში და ქვედა შასები შეიძლება დაცარიელდეს ან შეიძლება ჰქონდეს IOMX მოდული, როგორც ესაა მოცემული ამ მაგალითში.[5]

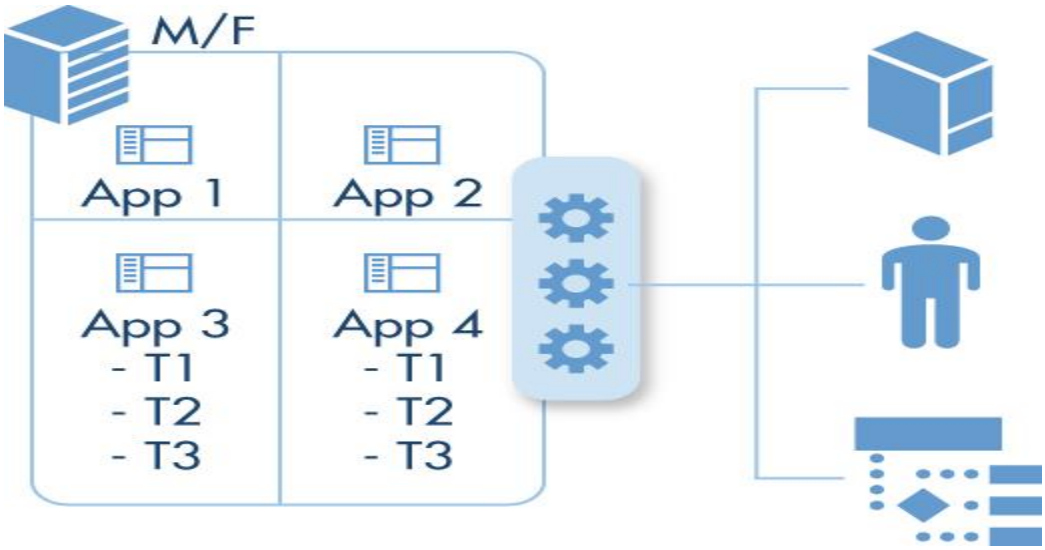


[5] https://docs.netapp.com/ontap-9/index.jsp?topic=%2Fcom.netapp.doc.dot-cm-sanconf%2FGUID-9CD2611E-7081-4125-8418-D223C407B86E.html

აპლიკაციის კონსოლიდაცია

უმეტესობა მსხვილ საწარმოში, როგორც წესი, არსებობს მთელი რიგი პროგრამების, რომლებსაც აქვს დუბლირების ფუნქცია. არსებობს ორი ბიზნეს პრობლემა, რომელიც გამომდინარეობს ამ სიტუაციიდან. ერთ დონეზე, შეიძლება არსებობდეს განსხვავებები სხვადასხვა სისტემებზე ინფორმაციის დამუშავების პროცესში, რამაც შეიძლება გამოიწვიოს პრობლემები ოპერაციებში ან ფინანსურ ანგარიშგებაში. საკმაოდ ძვირი ჯდება არსებითად ერთიდაიგივე რამდენიმე სისტემის შენარჩუნება.

ეფექტიანი დაგეგმარების მართვის პროცესი გაძლევთ შესაძლებლობას, მოხდინოთ ამ გადაჭარბებული აპლიკაციების კონსოლიდირება, მომსახურების ხარჯების შესამცირებლად. ოპტიმალურად, ყველა დუბლირებული აპლიკაცია შეიძლება დაყვანილი იქნას ერთიან აპლიკაციაზე შედარებით შემცირებული საერთო მომსახურებით, რომელიც ხდება ბიზნეს-პროცესების ფუნქციონირებისათვის ერთადერთი წყარო.



Akana's Portfolio Manager - ეს არის იმ მმართველობითი დაგეგმვის ერთადერთი გზა, რომელიც ორიენტირებულია საწარმოს მომსახურების დაგეგმვაზე. ეს საშუალებას აძლევს არქიტექტორებს იმუშაონ ინტენსიურად ბიზნეს ანალიტიკოსებთან და მმართველ მენეჯერებთან მმართველობის დაგეგმვის დროს. სხვადასხვა დაინტერესებულ მხარეებს შეუძლიათ "შექმნან" სასურველი საწარმოო მომსახურების ისეთი მდგომარეობა, რომლებიც არსებობს "როგორც არის" და დაგეგმონ და მიანიჭონ პრიორიტეტები მომსახურების საგზაო რუკაში. პორტფელის მენეჯერი უზრუნველყოფს მართვის დაგეგმვის საფუძველს, რომელიც

აერთიანებს ბიზნეს მოთხოვნებს "ზემოდან-ქვემოთ" და აპლიკაციების “ინვენტარიზაციას“ ქვემოდან-ზემოთ, წარმოადგენს რა პროიოტრიტეტულ ბიზნეს-მოთხოვნებს მომსახურების კანდიდატებად, რომლებიც მიიღება ამ სახეების ზემოდან-ქვემოთ და ქვემოდან-ზემოთ გადაკვეთით. დაგეგმვის პროცესის შედეგს წარმოდგენს საწარმოო მომსახურების საგზაო რუკის გამოქვეყნება, რომელიც შეიცავს ცალკეული სერვერ-კანდიდატების მომსახურების განსაზღვრებებს.

Akana- ის საერთო პოლიტიკის ინფრასტრუქტურის ძირითადი უპირატესობა - კომპანიის უნიფიცირებული ერთიანი პროდუქტის ნაკრების განსაკუთრებულობა - ეს არის შესაძლებლობა, უზრუნველყოს პოლიტიკის თანმიმდევრულობა იმ მთელი საწარმოო მომსახურების სასიცოცხლო ციკლის განმავლობაში, რომელიც დაგეგმვაზეა დაფუძნებული. იქიდან გამომდინარე, რომ დაგეგმვის პროცესიდან გამომავალი მონაცემები გადადიან დამუშავების და SOA Governance-ის ექსპლუატეციის ეტაპზე, გამჭირვალე უსაფრთხოების და SLA-ს პოლიტიკა რომელიც დაკავშირებულია დაგეგმის ეტაპზე Policy Manager-ით განსაზღვრული მომსახურების კანდიდატთან, ადვილად გადადიან შემუშავებაზე, სადაც მართვა ხორციელდება Akana’s Repository Manager-ის მიერ. როდესაც საწარმოო მომსახურება განლაგებულია, ამ პოლიტიკის განხორციელება ხდება Akana- ის მომსახურების მენეჯერის მიერ.[6]

როგორც ღრუბლოვანი ტექნოლოგიების სპეციალისტები აღნიშნავენ, IT-ინფრასტრუქტურის კონსოლიდაცია გვევლინება პირველ ნაბიჯად „ღრუბელში“. იმისათვის რომ გადავიდეთ ღრუბლოვანი ტექნოლოგიების გამოყენებაზე, კომპანიებმა თავდაპირველად უნდა გადაჭრან IT ტექნოლოგიის არაკონსოლიდირებულობის ამოცანა. კონსოლიდაციის გარეშე შეუძლებელია ავაგოთ ეფექტური პროცესზე-ორიენტირებული მართვა, რამდენადაც არ არსებობს სერვისების ერთიანი წარმოდგენის წერტილი.

ინფორმაციული ტექნოლოგიების განვითარების ისტორიის და თანამედროვე ტენდენციების გაანალიზების შედეგად შეიძლება გავაკეთოთ დასკვნა, რომ *IT*-ს ევოლუციური სპირალი, რომელიც მეინფრეიმების ეპოქასთან ერთად დაიწყო, შეიკრა. ღრუბლებთან ერთად ჩვენ დავუბრუნდით რესურსების ცენტრალიზაციას, მაგრამ ამჯერად არა მეინფრეიმების და მათი მწვანე ტერმინალების არამედ ახალ ტექნოლოგიურ დონეზე.

[6] https://www.akana.com/solutions/application_consolidation

თემა 5. ვირტუალიზაციის ტექნოლოგია

ვირტუალიზაცია ღრუბლოვან გამოთვლებში

ვირტუალიზაციია - ეს არის "რაღაცის ვირტუალური (უფრო სწორედ ფაქტობრივი) ვერსიის შექმნა, როგორიცაა სერვერი, დესკტოპი, შენახვის მოწყობილობა, ოპერაციული სისტემა ან ქსელის რესურსები".

სხვა სიტყვებით რომ ვთქვათ, ვირტუალიზაცია - ეს არის მეთოდი, რომელიც საშუალებას იძლევა, ერთობლივად გამოყენებული იქნას რესურსის ერთი ფიზიკური ინსტალაცია ან აპლიკაცია რამდენიმე სხვადასხვა მომხმარებლებსა და ორგანიზაციებს შორის. ეს ხდება ფიზიკური საცავზე ლოგიკური სახელის მინიჭების გზით და მოთხოვნისას ამ ფიზიკურ რესურსზე მითითებით.

რას წარმოადგენს ვირტუალიზაციის კონცეფცია?

ვირტუალური მანქანების შექმნა არსებული ოპერაციული სისტემის და აპარატურის მეშვეობით ცნობილია როგორც მოწყობილობის ვირტუალიზაცია. ვირტუალური მანქანა უზრუნველყოფს გარემოს, რომელიც ლოგიკურად გამოყოფილია ძირითადი აპარატურისგან.

მანქანა, რომელზეც ვირტუალური მანქანა შეიქმნება, ცნობილია როგორც ჰოსტ მანქანა (**Host Machine**) და ვირტუალური მანქანა არის გესტ მანქანა (**Guest Machine**).

ვირტუალიზაციის სახეებია:

1. აპარატურული ვირტუალიზაცია.
2. ოპერაციული სისტემის ვირტუალიზაცია.
3. სერვერების ვირტუალიზაცია.
4. საცავის ვირტუალიზაცია.

1) აპარატურული ვირტუალიზაცია:

როდესაც ვირტუალური მანქანა, პროგრამული უზრუნველყოფა ან ვირტუალური მანქანების მენეჯერი (VMM) პირდაპირ აპარატორულ სისტემაშია დაინსტალირებული, ამას ეწოდება აპარატორული ვირტუალიზაცია.

ჰიპერვიზორის მთავარ ამოცანას წარმოადგენს პროცესორის, მეხსიერების და სხვა აპარატურის რესურსების კონტროლი და მონიტორინგი.

აპარატურის ვირტუალიზაციის შემდეგ ჩვენ შეგვიძლია მასზე დავაინსტალიროთ სხვადასხვა ოპერაციული სისტემა და გავუშვათ სხვადასხვა აპლიკაციები ამ OS- ზე.

გამოყენება:

აპარატურული ვირტუალიზაცია ძირითადად ხდება სერვერის პლატფორმებისთვის, რადგან ვირტუალური აპარატების კონტროლი ბევრად უფრო ადვილია, ვიდრე ფიზიკური სერვერის მართვა.

2) ოპერაციული სისტემის ვირტუალიზაცია:

როდესაც ვირტუალური მანქანის პროგრამული უზრუნველყოფა ან ვირტუალური მანქანა მენეჯერი (VMM) დამონტაჟებულია ჰოსტ ოპერაციული სისტემაში, და არა უშუალოდ აპარატურის სისტემაში, ამას ეწოდება ოპერაციული სისტემის ვირტუალიზაცია.

გამოყენება:

ოპერაციული სისტემის ვირტუალიზაცია ძირითადად გამოიყენება OS-ის სხვადასხვა პლატფორმებზე აპლიკაციების შესამოწმებლად.

3) სერვერების ვირტუალიზაცია:

როდესაც ვირტუალური მანქანის პროგრამული უზრუნველყოფა ან ვირტუალური მანქანა მენეჯერი (VMM) დამონტაჟებულია უშუალოდ სერვერის სისტემაში, ამას ეწოდება სერვერის ვირტუალიზაცია.

გამოყენება:

სერვერების ვირტუალიზაცია იმიტომ კეთდება, რომ მოთხოვნის საფუძველზე და დატვირთვის ბალანსებისათვის ერთი ფიზიკური სერვერი შეიძლება დავყოთ რამდენიმე სერვერად.

4) საცავის ვირტუალიზაცია:

საცავის ვირტუალიზაცია - ეს არის რამდენიმე ქსელური საცავი მოწყობილობიდან ფიზიკური საცავის დაჯგუფების პროცესი ისე, რომ იგი გამოიყურებოდეს როგორც ერთი საცავი მოწყობილობა.

საცავის ვირტუალიზაცია ასევე ხორციელდება პროგრამების გამოყენებით.

გამოყენება:

საცავის ვირტუალიზაცია ძირითადად კეთდება რეზერვირებისა და აღდგენის მიზნით.

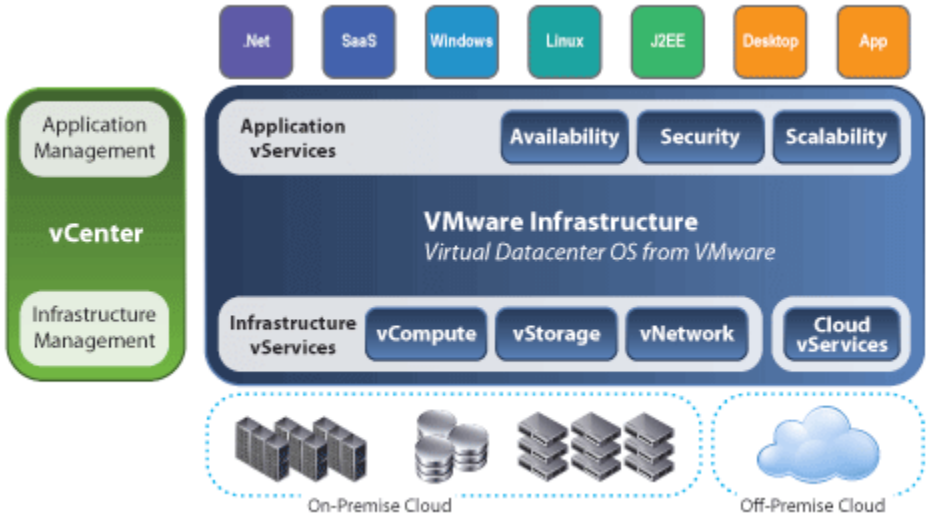
როგორ მუშაობს ვირტუალიზაცია ღრუბლოვან გამოთვლებში?

ვირტუალიზაცია მნიშვნელოვან როლს ასრულებს Cloud Computing ტექნოლოგიებში, ჩვეულებრივ, Cloud Computing-ში, მომხმარებლები ერთმანეთთან ცვლიან ღრუბლებში

არსებულ მონაცემებს, როგორიცაა, მაგალითად, აპლიკაცია და ა.შ. მაგრამ სინამდვილეში ვირტუალიზაციის დამხმარებით მომხმარებლები უზიარებენ ერთმანეთს ინფრასტრუქტურას.

ვირტუალიზაციის ტექნოლოგიის ძირითადი გამოყენება იმაში მდგომარეობს, რომ მათ ღრუბლოვან მომხმარებლებს წარმოუდგინონ აპლიკაციების სტანდარტული ვერსიები. ვარაუდობენ, რომ ამ ვერსიის მომდევნო ვერსია თუ იქნება გამოშვებული, მაშინ ღრუბლოვანმა პროვაიდერმა უნდა უზრუნველყოს უახლესი ვერსიის წარდგენა თავიანთი ღრუბლოვანი მომხმარებლებისდმი, რაც შედარებით ძვირია.

ამ პრობლემის გადასალახავად ვიყენებთ ვირტუალიზაციის იმ ძირითადად ტექნოლოგიას, რომელსაც იყენებს ვირტუალიზაცია. ყველა სერვერი და პროგრამული უზრუნველყოფა რომელიც სხვა ღრუბლოვან პროვაიდერებს მოითხოვს, მხარდაჭერილია მესამე მხარის მიერ და ღრუბლოვანი პროვაიდერებმა ყოველთვიურად ან ყოველწლიურად უნდა გადაიხადონ საფასური.



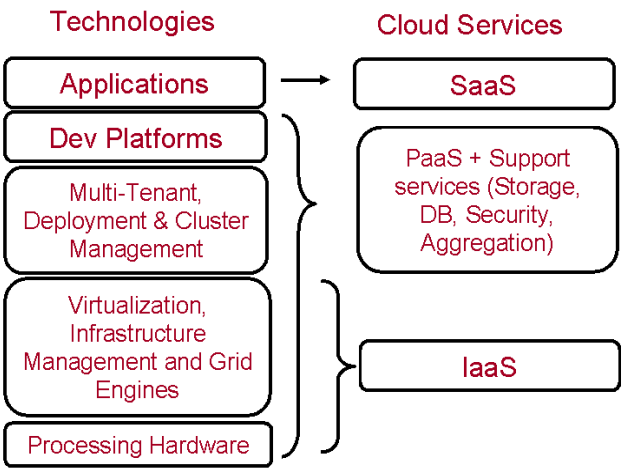
დასკვნა

ძირითადად ვირტუალიზაცია ნიშნავს, რამოდენიმე ოპერაციული სისტემის ერთი მანქანაზე გაშვებას, რომელიც ერთობლიობაში მოიხმარს ტექნიკის ყველა რესურს. ის გვეხმარება, წარმოვადგინოთ IT რესურსების აუზი, რათა ჩვენ შეგვიძლოს ერთობლივად მოვიხმაროთ ეს IT -რესურსები იმისათვის, რომ მიიღოთ სარგებელი ბიზნესში.

ვირტუალზიაციის ტექნოლოგია

ღრუბლოვანი კომპიუტერული სისტემის შესახებ ბევრი გაურკვევლობაა. კერძოდ, ღრუბელზე დაფუძნებულ მომსახურებებს (მაგ., SaaS, პროგრამული უზრუნველყოფა, როგორც

მომსახურება) და მოთხოვნით შეთავაზებული გამოთვლით ინფრასტრუქტურას შორის განსხვავება (IaaS, ინფრასტრუქტურა, როგორც მომსახურება). ქვემოთ მოყვანილი დიაგრამა ცდილობს გვიჩვენოს თუ როგორ აისახება ვირტუალიზაციის ტექნოლოგიები და ღრუბლოვანი გამოთვლების სერვისები რუკაზე.



Cloud Computing- ის ზოგადად მიღებული განმარტება არის სწრაფი და მასშტაბური, ვირტუალური გამოთვლითი რესურსების უზრუნველყოფა და მართვა, რომელის სარგებლობისათვის გადასახადის ოდენობა დამოკიდებულია ბრაუზერის გამოყენებზე, ზოგადად ინტერნეტით რესურსების მომხმარებლებთან დაკავშირებაზე. მაგრამ რესურსების ვირტუალიზაციისას მომხმარებელმა აღარ იცის, სად რა არის გაშვებული, ან როგორია პროცესერების საცავებთან სიახლოვე. ეს გავლენას ახდენს უსაფრთხოებაზე, გამტარუნარიანობასა და კონტროლის მოთხოვნებზე - აი სად შეუძლიათ ტელეკომუნიკაციურ კომპანიებს დახმარება, რადგან მათ შეუძლიათ დახმარება გაუწიონ ამ პრობლემის გადაწყეტაში და შეთავაზონ სრული პაკეტი მარტივად დოზირებულობის საფუძველზე.

მომწოდებლების ზოგიერთი მაგალითი მოიცავს:

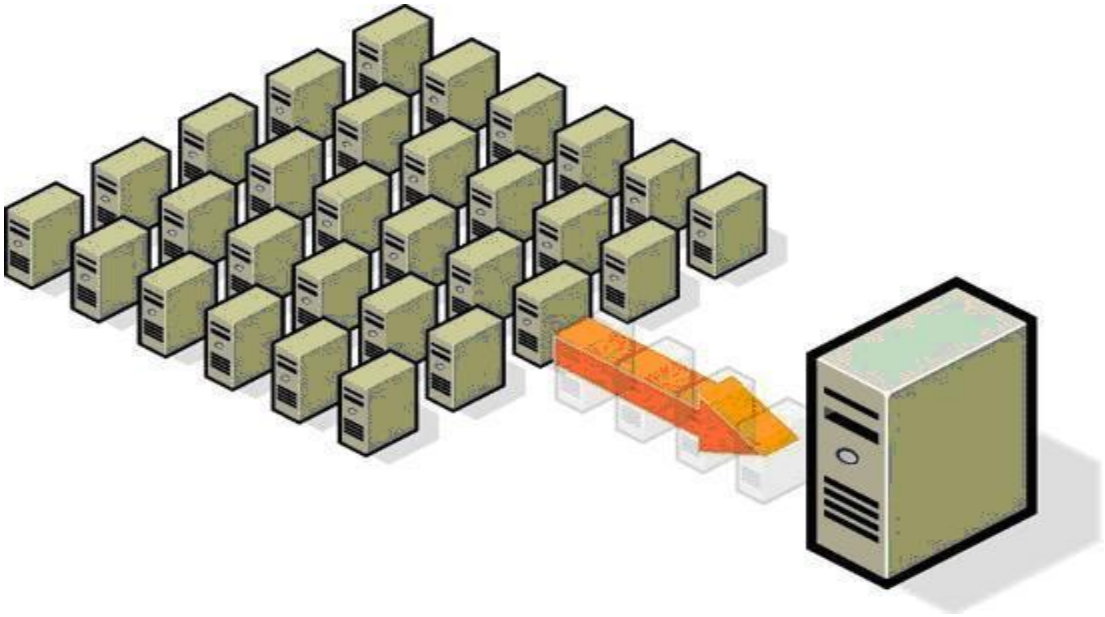
- ვირტუალიზაცია, ინფრასტრუქტურის მართვა და ქსელის ტექნოლოგიური ტექნოლოგიები: 3Tera, Cassatt, Citrix, Eucalyptus, Hadoop, IBM, Microsoft, Novel, Oracle (Sun), Parallels, Red Hat, Virtual Iron,VMWare.

• მრავალმომხმარებლიანი ტექნოლოგიები, განლაგებისა და კასეტური მართვის ტექნოლოგიები: 3Tera, Appistry, Elastra, Enomaly, Ecualyptus, Globus ალიანსი, Hadoop, Oracle (Sun), გამოთვლითი პლატფორმები, Q- Layer, RightScale.

• IaaS მიმწოდებლები: Akamai, AT & T, Amazon, Citirx, ElasticHosts, Flexiscale, Globus ალიანსი, GoGrid, Joyent, Layered Technologies,, Microsoft, Rackspace, Soasta, Verizon.

თუმცა, რამდენადაც მომხმარებელი იწყებს Cloud მომსახურებაზე დაეყრდნობას ჩვენ ვაწყდებით რიგი წარუმატებლობს.[7]

ფიზიკური სერვერის *ვირტულაიზაცია* საშუალებას იძლევა იგი მოქნილად განაწილდეს აპლიკაციებს შორის, ამასთან თითოეული „ხედავს“ მხოლოდ მისთვის განკუთვნილ რესურსებს და „თვლის“, რომ მისთვის გამოყოფილია ცალკე *სერვერი*, ანუ მოცემულ შემთხვევაში რეალიზდება მიდგომა „ერთი სერვერი - რამოდენიმე აპლიკაცია“, ოღონდ სერვერული აპლიკაციის მწარმოებლურობის, შეღწევადობის და უასფრთხოების შემცირების გარეშე. გარდა ამისა, ვირტუალიზაციის გადაჭრა საშუალებას იძლევა გაიშვას განყოფილებებში სხვადასხვა ოპერაციულ სისტემა მათი სისტემური სერვერის აპარატულ რესურსებისადმი მიმართვების სიმულაციის მეშვეობით.

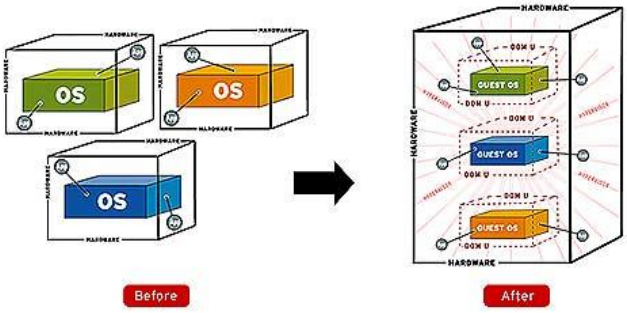


ნახ. 11. ვირტუალზიაცია გულისხმობს ერთ ფიზიკურ კომპიუტერზე რამდენიმე ვირტუალური კომპიუტერის გაშვებას.

ტექნოლოგია, რომელიც Cloud Computing-ს შესაძლებელს ხდის - ეს არის ვირტუალიზაცია. Cloud Computing-ში მთავარი მიზანს წარმოადგენს, რესურსების

[7] http://alanquayle.com/2009/10/virtualization-technology-and/

გამოყენების გაუმჯობესება, საჭიროების მიხედვით, რამდენიმე მომხმარებლს შორის მოთხოვნილი რესურსების გაზიარებით. ვირტუალიზაციის ახდენს ფუძემდებლური რესურსების აბსტრაქტიზაციას, როგორიცაა მეხსიერება, საცავი, ქსელი, ისე, რომ რამდენიმე ოპერაციულ სისტემას (Windows, Linux) შეუძლიათ ერთდროულად იმუშაონ ერთ ფიზიკურ სისტემაზე. ეს აუმჯობესებს რესურსების გამოყენებას - ტრადიციული სერვერები სარგებლობს დაახლოებით 20% -იანი რესურსებით, ხოლო ვირტუალური მანქანა (ეს არის აღნიშნული ინდივიდუალური ოპერაციული სისტემა, რომელიც ზემოთაა მოხსენიებული) იყენებს მისი რესურსების საშუალოდ 80% -ს.



ღრუბელი სამომავლო სფეროა ტექნოლოგიის მიწოდებისათვის. დღეს, ღრუბლოვანი მომსახურების კომპანიები ათეულობით არიან - ისინი ყოველნაირად ცდილობენ cloud ბაზრის დიდი ნაწილის ხელში ჩაგდებას. ვირტუალიზაციის უპირატესობები სცილდება მხოლოდ რესურსების გამოყენებას - ვირტუალიზებული სერვერები ნაკლებ ენერგიას მოიხმარენ, აქვთ გაგრილების ნაკლები ხარჯები და სჭირდებათ ნაკლები სივრცე (გარემოსთვის ყველაფერი ძალიან კარგია). ვირტუალიზაციის კი - ეს არის ძირითადი ტექნოლოგია, რომელიც უზრუნველყოფს რესურსების სწრაფ გამოყოფას, მოთხოვნით რესურსით სარგებლობას, კომუნალურ მომსახურების ღირებულების და კაპიტალური ხარჯების შემცირების საშუალებას იძლევა.[8]

ვირტუალზიაციის საფუძველში დევს იმის შესაძლებლობა, რომ ერთმა კომპიუტერმა შეასრულოს რამოდენიმე კომპიუტერის სამუშაო მისი რესურსების რამოდენიმე გარემოში განაწილების დახმარებით. ვირტუალური სერვერების და ვირტუალური სამაგიდო კომპიუტერების საშუალებით შესაძლებელია განთავსდეს რამოდენიმე ოპერაციული სისტემა და აპლიკაცია ერთ ადგილზე. ამგვარად, ფიზიკურ და გეოგრაფიულ შეზღუდვას აღარ გააჩნია არანაირი მნიშვნელობა. აპარატურლი რესურსების ეფექტური გამოყენებით მიღებული

[8] https://bluelock.com/blog/virtualization-the-cloud-computing-enabler/

ენერგოდაზოგვისა და ხარჯების შემცირების გარდა, ვირტუალური *ინფრასრუქტურა* უზრუნველყოფს რესურსების ხელმისაწვდომობის მაღალ დონეს, უფრო ეფექტურ მართვის სისტემას, გაზრდილ უსაფრთხოებას და კრიტიკულ სიტუაციაში აღდგენის სრულყოფილ სისტემას.

ფართე გაგებით ვირტუალიზაცია წარმოადგენს რომელიმე პროცესის ნამდვილი რეალიზაციის ან ობიექტის დამალვას იმათთვის, ვინც მათ იყენებს. ვირტუალიზაციის პროდუქტია ისეთი რამაა, რაც მოსახერხებელია გამოსაყენებლად, ფაქტობრივად გააჩნია ობიექტზე მუშაობისას აღქმადზე უფრო რთული და განსხვავებული სტრუქტურა. სხვა სიტყვებით, ხდება წარმოდგენის გამოყოფა რაიმეს რეალიზაციისგან. *ვირტუალიზაცია* მოწოდებულია გამოყოს პროგრამული უზრუნველყოფა აპარატული ნაწილისაგან.

კომპიუტერულ ტქნოლოგიებში ტერმინი „ვირტუალიზაციის“ ქვეშ ჩვეულებრივ იგულისხმება გამომთვლელი რესურსების აბსტრაქცია და მომხმარებლისთვის სისტემის წარმოდგენა, რომელიც „ინკაპსულირებს“ *(თავის თავში მალავს)* საკუთარ რეალიზაციას. მარტივად რომ ვთქვათ, *მომხმარებელი* მუშობს მისთვის მოსახერხებელ წარმოდგენილ ობიექტთთან და არ აქვს მისთვის მნიშვნელობა რეალურად როგორაა ობიექტი მოწყობილი.

რამოდენიმე ვირტუალური მანქანის გაშვება ერთ ფიზიკურ კომპიუტერზე დიდ ინტერესს იწვევს კომპიუტერულ სპეციალისტებს შორის, არა მარტო იმიტომ, რომ ეს ზრდის IT-ინფრასტრუქტურის მოქნილობას, არამედ იმიტომაც, რომ ვირტუალზიაცია სინამდვილეში ფულის დაზოგვის საშუალებას იძლევა.

ვირტუალიზაციის უპირატესობა

ვირტუალიზაცია ჩნდება მაშინ, როდესაც ვირტუალური ვერსია იქმნება ფაქტობრივი ვერსიის ნაცვლად. თანამედროვე ვირტუალიზაციაში ეს შეიძლება იყოს საცავის მოწყობილობები, ქსელები, ოპერაციულ სისტემები ან სერვერებიც კი. ეს არის პროცესი, რომელიც დაიწყო 1960-იან წლებში, როდესაც ზოგიერთმა ადრეულმა მეინფრეიმ-კომპიუტერებმა გაანაწილეს თავიანთი სისტემური რესურსები ერთდროულად სხვადასხვა აპლიკაციებზე მუშაობისთვის.

ამის შემდეგ, ვირტუალიზაციამ მოიცვა ციფრული ცხოვრების პრაქტიკულად თითქმის ყველა ფორმა. ვირტუალური მანქანების გამოყენებით, რომლებიც მოქმედებენ როგორც

რეალურები კომპიუტერები და რომელიც ემყარება ემულაციას (*კომპიუტერის ან ელექტრონული მოწყობილობის მიერ გადაპროგრამების გარეშე სხვა / განსხვავებული ფუნქციური შესაძლებლობების მქონე კომპიუტერის ან ელექტრონული მოწყობილობის მუშაობის იმიტაცია)*, ბევრი ადამიანი ისარგებლებს ვირტუალიზაციის უპირატესობით.

უმეტეს ტექნოლოგიების მსგავსად, ვირტუალიზაციას გააჩნია უპირატესობები და უარყოფითი მხარეები, რომლებიც გათვალისწინებული უნდა იყოს სისტემის ან გეგმის სრულად განხორციელებამდე.

ვირტუალიზაციის უპირატესობები

1. ეს არის იაფი.

იმის გამო, რომ ვირტუალიზაციას არ საჭიროებს რეალური აპარატურული კომპონენტების გამოყენებას ან დამონტაჟებას, IT ინფრასტრუქტურები მიიჩნევენ მას, როგორც შედარებით იაფი სისტემას დანერგვისათვის. აღარ არის საჭირო დიდი ფართობის გამოყოფა და უზარმაზარი ფულადი ინვესტიციები ადგილზე რესურსების შესაქმნელად. თქვენ უბრალოდ იძინეთ ლიცენზიას ან ხელმისაწვდომობას მესამე მხარის პროვაიდერიდან და იწყებთ მუშაობას, თითქოს აპარატი დამონტაჟდა ადგილობრივად.

2. ზოგავს წინასწარ განსაზღვრულ ხარჯებს.

იმის გამო, რომ გარეშე პროვაიდერები, როგორც წესი, ვირტუალიზაციის ვარიანტებს სთავაზობენ, კერძო პირებს და კორპორაციებს შეიძლება გააჩნდეთ წინასწარ განსაზღვრული ხარჯები მათი საინფორმაციო ტექნოლოგიების საჭიროებებისათვის.

3. ამცირებს დატვირთვას.

ვირტუალიზაციის პროვაიდერების უმეტესობა ავტომატურად განაახლებს თავის აპარატურას და პროგრამულ უზრუნველყოფას, რომლებიც გამოყენებული იქნება. იმის ნაცვლად, რომ გაგზავნონ ხალხი ამ განახლებებისათვის, ისინი ყენდება გარე მომწოდებლის მიერ. ეს საშუალებას აძლევს ადგილობრივი IT სპეციალისტებს ფოკუსირება მოახდინონ სხვა ამოცანებზე და ამით ფიზიკური პირებისათვის და ან კორპორაციებისათვის კიდევ უფრო მეტი თანხა დაიზოგონ.

4. ის გთავაზობთ უკეთეს uptime-ს (კალენდარული დრო, რომლის განმავლობაში სისტემას შეუძლია შეასრულოს აუცილებელი ფუნქციები).

ვირტუალიზაციის ტექნოლოგიების წყალობით, uptime მნიშვნელოვნად გაუმჯობესდა. ზოგიერთი პროვაიდერები გთავაზობთ uptime-ს რომელიც 99,9999%-ს შეადგენს. ბიუჯეტთან მეგობრული პროვაიდერებიც კი დღეს გთავაზობენ uptime-ის 99.99%-ს.

5. ის საშუალებას იძლევა სწრაფად განახლდეს რესურსები.

ვირტუალიზაციის გამოიყენებისას რესურსებით უზრუნველყოფა სწრაფად და მარტივად ხდება. აღარ არის საჭირო ფიზიკური აპარატების შექმნა, ადგილობრივი ქსელების შექმნა ან სხვა საინფორმაციო ტექნოლოგიების კომპონენტების დაყენება. სანამ ვირტუალური გარემოზე ხელმისაწვდომობის მინიმუმ ერთი წერტილი არსებობს, ის შეიძლება გავავრცელოთ მთელ დანარჩენ ორგანიზაციაზე.

6. ხელს უწყობს ციფრულ მეწარმეობას.

სანამ ფართომასშტაბიანი ვირტუალიზაცია მოხდა, ციფრული მეწარმეობა პრაქტიკულად შეუძლებელი იყო ჩვეულებრივი ადამიანისათვის. სხვადასხვა პლატფორმების, სერვერებისა და საცავი მოწყობილობების წყალობით, რომლებიც დღეს ხელმისაწვდომია, თითქმის ყველას შეუძლია დაიწყოს საკუთარი საქმიანობა ან გახდეს ბიზნესის მფლობელი. ისეთი საიტები, როგორიცაა Fiverr და UpWork ნებისმიერს აძლევს საშუალებას შექმნას სანიშნე და დაიწყოს ზოგიერთი სამუშაოს მოძიება.

7. უზრუნველყოფს ენერგიის დაზოგვას.

უმეტესობა კერძო პირისა და კორპორაციებისათვის ვირტუალიზაცია არის ენერგოეფექტური სისტემა. იმის გამო, რომ გამოიყენება ადგილობრივი აპარატორული ან პროგრამული უზრუნველყოფის პარამეტრები, ენერგომოხმარება შეიძლება შემცირდეს. მონაცემთა ცენტრის გაგრილების და აღჭურვილობის საოპერაციო ხარჯების გადახდის ნაცვლად, ფულადი სახსრები შეიძლება გამოყენებულ იქნას სხვა ოპერაციულ ხარჯებზე, რათა მოხდეს ვირტუალიზაციის საერთო ROI- ს გაუმჯობესება.

ვირტუალიზაციის ნაკლოვანებები

1. მას შეიძლება ჰქონდეს დანერგვის მაღალი ღირებულება.

საშუალო კერძო პირისათვის ან ბიზნესისათვის ღირებულება, როდესაც ვირტუალიზაციის საკითხი განიხილება, საკმაოდ დაბალია. მაგრამ ვირტუალიზაციის გარემოს მომწოდებელებისათვის დანერგვის ხარჯები შეიძლება საკმაოდ მაღალი იყოს. აპარატურა და პროგრამული უზრუნველყოფა საჭიროა რაღაც მომენტში და ეს იმას ნიშნავს,

რომ მოწყობილობები ან უნდა იყოს შემუშავებული, დამზადებული ან შეძენილი განხორციელებისათვის.

2. მას აქვს შეზღუდვები.

ყველა აპლიკაცია ან სერვერი არ მუშაობს ვირტუალიზაციის პირობებში. ეს იმას ნიშნავს, რომ კერძო პირს ან კორპორაციას შეიძლება დასჭირდეს ჰიბრიდული სისტემა სწორად ფუნქციონირებისათვის. ეს ჯერჯერობით დროსა და ფულს ზოგავს გრძელვადიან პერსპექტივაში, მაგრამ იმის გამო, რომ ყველა გამყიდველი არ უჭერს მხარს ვირტუალიზაციას და ზოგიერთს შეუძლია შეწყვიტოს მხარდაჭერა მისი თავდაპირველად გაშვების შემდეგ, ყოველთვის არსებობს გაურკვევლობის დონე ამ ტიპის სისტემის სრული რეალიზაციის დროს.

3. ქმნის უსაფრთხოების რისკს.

ინფორმაცია ჩვენი თანამედროვე ვალუტაა. თუ თქვენ ის გაქვთ, შეგიძლიათ გააკეთოთ ფული. თუ თქვენ ის არ გაქვთ, თქვენ უარგყოფენ. იმის გამო, რომ მონაცემებს გადამწყვეტი მნიშვნელობა აქვს ბიზნესის წარმატებისთვის, ისინი ხშირად მიზანმიმართულია. 2017 წელს მონაცემთა უსაფრთხოების დარღვევის საშუალო ღირებულება, პონტონის ინსტიტუტის მიერ გამოქვეყნებული დასკვნის თანახმად, იყო 3.62 მილიონი დოლარი. პერსპექტივა: მეხის დარტყმის ალბათობა დაახლოებით 1 მილიონზე ერთს შეადგენს. ვირტუალიზაციის გამოყენებისას მონაცემთა დარღვევის შანსები? ერთი ოთხთანაა.

4. ქმნის ხელმისაწვდომობის პრობლემას.

ძირითადი პრობლემა, რომელიც ბევრს აქვს ვირტუალიზაციის სფეროში იმაში მდგომარეობს, რა მოუვა მათ სამუშაოს, თუ მათ არ ექნებათ აქტივებთან წვდომის შესაძლებლობა. თუ ორგანიზაცია ვერ შეძლებს თავიანთ მონაცემებთან დაკავშირებას ხანგრძლივი პერიოდის განმავლობაში, მაშინ მათ მოუწევთ თავიანთ დარგში კონკურენციასთან ბრძოლა. რამდენედაც ხელმისაწვდომობა კონტროლდება მესამე მხარის პროვაიდერების მიერ, კავშირზე დარჩენის შესაძლებლობა არ კონტროლდება ვირტუალიზაციის მიერ.

5. ქმნის სკალარულობის პრობლემას.

მიუხედავად იმისა, რომ ვირტუალიზაციის მეშვეობით შეიძლება სწრაფად განავითაროთ ბიზნესი ან შესაძლებლობები, თქვენ შეიძლება ვერ მიღწიოთ ისეთ დიდ შედეგს, როგორიც გსურთ. თქვენ ასევე შეიძლება მოითხოვოთ რომ ის თავიდანვე იყოს უფრო

დიდი. იმის გამო, რომ მრავალი კომპანია იყენებს ერთი და იმავე რესურსს, ზრდა ქმნის შუალედს ვირტუალიზაციის ქსელში. ერთი დიდის ყოფნას შეუძლია წაართვას რესურსების რამდენიმე მცირე ბიზნესს და არავის არ შეუძლია ამას გაუმკლავდეს.

6. საჭიროა რამდენიმე რგოლი ჯაჭვში, რომელებიც ერთობლივად უნდა თანამშრომლობდნენ.

თუ თქვენ გაქვთ ადგილობრივი აღჭურვილობა, მაშინ თქვენ სრულად აკონტროლებთ, თუ რისი გაკეთება შეგიძლიათ. ვირტუალიზაციის შემთხვევაში, თქვენ დაკარგავთ ამ კონტროლს, რამდენადაც ლინკმა ერთად უნდა ითანამშრომლონ იმავე ამოცანის შესასრულებლად. მოდით გამოიყენოთ დოკუმენტის ფაილის შენახვის მაგალითი. ადგილობრივი შენახვის მოწყობილობით, როგორცაა ფლეშ დრაივი ან HDD, შეგიძლიათ შეინახოთ ფაილი დაუყოვნებლივ და შქმნათ სარეზერვო ასლიც კი. ვირტუალიზაციის გამოყენებისა, თქვენი ISP კავშირი უნდა იყოს მოქმედი. თქვენი LAN ან Wi-Fi უნდა მუშაობდეს. თქვენი ონლაინ -საცავი უნდა იყოს ხელმისაწვდომი. იმ შემთხვევაში, თუ რომელიმე არ მუშაობს, მაშინ თქვენ ვერ შეინახავთ ამ ფაილს.

7. სჭირდება დრო.

თუმცა, თქვენ ზოგავთ დროს ვირტუალიზაციის დანერგვის ეტაპებზე, გრძელვადიან პერსპექტივაში მომხმარებელი მეტ დროს ხარჯავს ადგილობრივ სისტემებთან შედარებით. ეს იმიტომ ხდება, რომ არსებობს დამატებითი ნაბიჯები, რომელსაც უნდა მოჰყვეს სასურველი შედეგი.

ვირტუალიზაციის უპირატესობები და უარყოფითი მხარეები გვიჩვენებენ, რომ მისი სწორი გამოყენებით იგი შეიძლება იყოს სასარგებლო ინსტრუმენტი ფიზიკური პირებისათვის, SMBs- ისთვის, მეწარმეებისათვის და კორპორაციებისთვის. მაგრამ იმის გამო, რომ მისი გამოყენება ძალიან ადვილია, ზოგიერთი ადმინისტრატორი ამატებს ახალ სერვერებს ან საცავებს და ქმნის გაფართოვებას. თუ დარჩებიან კომუნიკაციის საკითხებზე დისციპლინირებული და გათვითცნობიერებულები, შესაძლებელია უარყოფითი მხარის გადალახვა, სწორედ ამის გამოც არის თანამედროვე სისტემა ასეთი ეფექტური.[9]

ოპერაციულ სიტემას არ შუძლია განასხვავოს ვირტუალური და ფიზიკური მანქანა. იგივე შეიძლება ითქვას აპლიკაციებზე და ქსელში სხვა კომპიუტერებზე. თვითონ

[9] https://vittana.org/14-advantages-and-disadvantages-of-virtualization

ვირტუალური მანქანაც კი თავის თავს თვლის როგორც „ნამდვილ“ კომპიუტერს. მაგრამ ამის მიუხედავად ვირტუალური მანქანები შედგება მხოლოდ პროგრამული კომპონენტებისაგან და არ შეიცავენ მოწყობილობებს. ეს მათ აძლევთ ფიზიკურ მოწყობილობებზე რიგ უნიკალურ უპირატესობას.

ნახ. 12. ვირტუალური მანქანა

განვიხილოთ ვირტუალური მანქანის თავისებურებანი უფრო დეტალურად:

1. **თავსებადობა.** ვირტუალური მანქანები, როგორც წესი, თავსებადია ყველა სტანდარტულ კომპიუტერთან. როგორც ფიზიკური კომპიუტერი, ვირტუალური მანქანა მუშაობს საკუთარი ჰოსტური ოპერაციული სისტემის მართვის ქეშ და ასრულებს საკუთარ აპლიკაციებს. ის ასევე შეიცავს ფიზიკური კომპიუტერისათვის ყველა სტანდარტულ კომპონენტს (სისტემური პლატას ანუ დედა პლატას, ვიდეო ბარათს, ქსელის კონტროლერს და ა.შ.). ამიტომ ვირტუალური მანქანები სრულად თავსებადია ყველა მოწყობილობების სტანდარტულ ოპერაციულ სისტემასთან, აპლიკაციებთან და დრაივერებთან. ვირტუალური მანქანა შეიძლება გამოყენებული იყოს ნებისმიერი პროგრამული უზრუნველყოფისათვის, რომელიც ვარგისია შესაბამისი ფიზიკური კომპიუტერისათვის.

2. **იზოლირებულობა**. ვირტუალური მანქანები, ისე როგორც ფიზიკური კომპიუტერები, სრულად იზოლირებულნი არიან ერთმანეთისაგან. ვირტუალურ მანქანებს შეუძლიათ გამოიყენონ ერთი კომპიუტერის საერთო ფიზიკური რესურსი და ამასთან დარჩნენ ერთმანეთისაგან სრულიად იზოლირებულნი ისე, როგორც ცალკეული ფიზიკური მანქანები. მაგალითად, თუ ერთ სერვერზე გაშვებულია ოთხი ვირტუალური მანქანა, და ერთ-ერთი მათგანი მწყობრიდან გამოვიდა, ეს არ მოახდენს გავლენას დანარჩენი სამი მანქანის ხელმისაწვდომობაზე. იზოლირებულობა – ეს არის აპლიკაციების უფრო მაღალი ხელმისაწვდომობის და უსაფრთხოების მნიშვნელოვანი მიზეზი, რომლებიც სრულდება ვირტუალურ გარემოში, იმ აპლიკაციებთან შედარებით, რომლებიც სრულდება სტანდარტულ, არავირტუალურ სისტემაში.

3. **ინკაპსულაცია**. ვირტუალური მანქანები სრულად ახდენენ გამოთვლითი გარემოს ინკაპსულაციას. ვირტუალური მანქანა წარმოადგენს პროგრამულ კონტეინერს, რომელიც არის ვირტუალური აპრატურული რესურსების სრული კომპლექტის, ასევე ოპერაციული სისტემის და პროგრმულ პაკეტში არსებული მისი ყველა აპლიკაციის დამაკავშირებელი, ანუ „ინკაპსულირებელი“. ინკაპსულაციის წყალობით ვირტუალური მანქანები ხდებიან წარმოუდგენლად მობილურები და მოსახერხებელნი მართავაში. მაგალითად, ვირტუალური მანქანის გადაადგილება ან კოპირება შეიძლება ერთი ადგილმდებარეობიდან მეორეში ისევე როგორც ნებისმიერი პროგრამული ფაილის. გარდა ამისა ვიტუალური მანქანის შენახვა შეიძლება ნებისმიერ მონაცმთა სტანდარტულ მატარებელზე: USB Flash-მეხსირების კომპაქტური ბარათიდან მონაცემთა შენახვის კორპორატიულ ქსელემდე.

4. **მოწყობილიბებისაგან დამოუკიდებლობა**. ვირტუალური მანქანები სრულიად დომოუკიდებელნი არიან იმ საბაზო ფიზიკური მოწყობილობებისგან, რომელზედაც ისინი მუშაობენ. მაგალითად, ვირტუალური კომპონენტების (ცენტრალური პროცესრორის, ქსელური ბარათის, SCSI კონტროლერების) მქონე ვირტუალური მანქანას შეიძლება მივცეთ პარამეტრები, რომელიც აბსოლუტურად არ ემთხვევა საბაზისო აპარატული უზრუნველყოფის ფიზიკურ მახასითებლებს. ვირტუალურ მანქანებს ასევე შეუძლიათ შეასრულონ სხვადასხვა ოპერაციული სისტემები (linux, windows და სხვა) ერთიდაიმავე ფიზიკურ სერვერზე. აპარატორული დამოუკიდებლობა ინკაპსულაციის და თავსებადობის თვისებებთან შეხამებით უზრუნველყოფს შესაძლებლობას თავისუფლად გადავაადგილოთ ვირტუალუტი მანქანები *x*86 ბაზაზე ერთი კომპიუტერიდან მეორეზე, დრავერის მოწყობილობების ან ოპერაციული სისტემის შეუცვლელად. მოწყობილიბებისაგან დამოუკიდებლობა ასევე იძლევა შესაძლებლობას ერთ ფიზიკურ კომპიუტერზე გაშვებული იყოს აბსოლუტურად სხვადასხვა ოპერაციული სისტემები და აპლიკაცია.

არსებობს ვირტუალიზაციის ძირითადი ნაირსახეობები, კერძოდ:

- სერვერების ვირტუალიზაცია (სრული ვირტუალიზაცია და პარავირტუალიზაცია)
- ვირტუალიზაცია ოპერატციული სისტემების დონეზე,
- აპლიკაციების ვირტუალიზაცია,
- პრეზენტაციის ვირტუალიზაცია.

თემა 6. სერვერების ვირტუალიზაცია

რა არის სერვერების ვირტუალიზაცია?

სერვერული ვირტუალიზაცია ახდენს აბსტრაქციას სერვერული პროგრამული უზრუნველყოფიდან აპარატორულ უზრუნველყოფაზე სტუმარი / მასპინძელის საფუძველზე, რაც საშუალებას აძლევს მრავალ ვირტუალურ სერვერს იმუშაოს ფიზიკურ მოწყობილობაზე.

სერვერების ვირტუალიზაციის განმარტება

სერვერი ვირტუალიზაციია - ეს არის პროგრამული არქიტექტურა, რომელიც საშუალებას იძლევა სერვერის რამდენიმე ოპერაციული სისტემამ იმუშაოს, როგორც სტუმარმა მოცემულ ფიზიკურ სერვერ-მასპინძელზე. ფიზიკური მანქანიდან სერვერის პროგრამული უზრუნველყოფის ასეთი დაშორების დროს სერვერი ხდება "ვირტუალური მანქანა", რომელიც დაშორებულია ფიზიკური სიბრტყიდან - თუმცა სერვერი "ფიქრობს", რომ ის გაშვებულია სწორედ გამოთვლით და მეხსიერების რესურსებზე. სინამდვილეში ის მუშაობს სერვერული მოწყობილობის ვირტუალურ იმიტაციაზე.

რატომ ვიყენებთ სერვერების ვირტუალიზაციის?

სერვერების ვირტუალიზაცია IT რესურსების უფრო ეფექტურად გამოყენების საშუალებას იძლევა, ვიდრე ეს ადრე იყო შესაძლებელი. სერვერების ვირტუალიზაციის დაწყებამდე მონაცემთა ბაზაში ერთი და იმავე მონაცემთა ბაზის არსებობის შემთხვევაში მონაცემთა ცენტრში ზედმეტად ჰქონდათ გამოყენებული ტექნიკა. ვირტუალიზაციის საშუალებით შესაძლებელია გადავანაწილოთ სამუშაო დატვირთვები ვირტუალური მანქანებს შორის დატვირთვების შესაბამისად. ერთი და იმავე ფიზიკური სერვერს ასევე შეუძლია იმუშაოს რამდენიმე სერვერულ ოპერაციულ სისტემასთან და კონფიგურაციასთან, რაც უფრო ზრდის ეფექტურობას. სერვერების ვირტუალიზაცია საფუძველია Cloud Computing-სთვის და ჰიბრიდული IT-სთვის.

HPE სერვერების ვირტუალიზაციის პროდუქცია და მომსახურება

HPE გთავაზობთ სერვერების ვირტუალიზაციისა და საცავების ვირტუალიზაციისათვის გადაწყვეტილებების ფართო, ადაპტირებულ პორტფელს. სერვერების ვირტუალიზაციისთვის

HPE- სთან პარტნიორობა ნიშნავს, რომ თქვენ მიიღებთ სწრაფ განლაგებას, მასშტაბის შესრულებას, სერვერისა და შენახვის რესურსების ეფექტურ გამოყენებას, ასევე მაღალი ხელმისაწვდომობის მიღწევის შესაძლებლობას. HPE- ს სერვერების ვირტუალიზაციასთან ერთად HPE- ის შენახვის პლატფორმის კომბინირებისას შეგიძლიათ გაიზარდოთ თქვენი Tier-1- ის სისწრაფე, საიმედოობა და სკალაცია და მისი კრიტიკულად ამნიშვნელოვანი აპლიკაციები.

სერვერის კომპიუტერები - მანქანები, რომელზედაც განლაგებულია კომპიუტერული ქსელების მასპინძელი ფაილები და პროგრამები - უნდა იყვნენ ძლიერები. ზოგიერთს გააჩნია ცენტრალური დამამუშავების ერთეული (CPUs) რამდენიმე პროცესორთან ერთად, რომლებიც ამ სერვერებს საშუალებას აძლევს კომპლექსური ამოცანები შეასრულონ მარტივად. კომპიუტერის ქსელის ადმინისტრატორები ჩვეულებრივ გამოყოფენ თითოეულ სერვერს კონკრეტული აპლიკაციის ან ამოცანისათვის. ამ ამოცანებიდან ბევრი სხვებთან კარგად არ თავსდება - ყოველ მათგანს საკუთარი გამოყოფილი მანქანა სჭირდება. სერვერზე ერთი აპლიკაცია ასევე საშუალებას იძლევა უფრო ადვილად იქნას გამოვლენილი პრობლემა მათი წარმოქმნისთანავე. ეს არის მარტივი გზა კომპიუტერის ქსელის ტექნიკური გამარტივების თვალსაზრისით.[10]

თუმცა ამ მიდგომას გააჩნია რამდენიმე პრობლემა. ერთი არის ის, რომ იგი არ სარგებლობს თანამედროვე სერვერული კომპიუტერების სიმძლავრეებით. სერვერების უმრავლესობა მათი საერთო დამუშავების შესაძლებლობების მხოლოდ მცირე ნაწილს იყენებს. კიდევ ერთი პრობლემა მდგომარეობს იმაში, რომ როგორც კი კომპიუტერული ქსელი უფრო დიდი და უფრო რთული ხდება, სერვერები ბევრ ფიზიკურ სივრცეს იკავებენ. მონაცემთა გადამუშავების ცენტრი შეიძლება გადატვირთული იყოს სერვერების სტელაჟებით, რომლებიც ბევრ ენერგიას მოიხმარენ და მეტ სითბოს გამოიმუშავებენ.

სერვერების ვირტუალიზაცია ცდილობს ორივე ეს პრობლემა ერთი ხელის მოსმით მოაგვაროს. სპეციალურად შექმნილი პროგრამული უზრუნველყოფის გამოყენებით, ადმინისტრატორს შეუძლია გადააკეთოს ერთი ფიზიკური სერვერი მრავალ ვირტუალური მანქანად. თითოეული ვირტუალური სერვერი მოქმედებს, როგორც უნიკალური ფიზიკური მოწყობილობა, რომელსაც შეუძლია საკუთარი ოპერაციული სისტემის (OS) გაშვება.

[10] https://www.hpe.com/us/en/what-is/server-virtualization.html

თეორიულად, შეგიძლიათ შექმნათ საკმარისი ვირტუალური სერვერები, რომ გამოიყენოთ მანქანის მთელი გამოთვლითი სიმძლავრე, თუმცა პრაქტიკაში ეს ყოველთვის არ არის საუკეთესო იდეა.

ვირტუალიზაცია არ არის ახალი კონცეფცია. კომპიუტერული მეცნიერები ათწლეულების განმავლობაში სუპერკომპიუტერებზე ვირტუალურ მანქანებს ქმნიდნენ. მაგრამ ეს მხოლოდ რამდენიმე წელია, რაც ვირტუალიზაციისთვის შესაძლებელი გახდა სერვერების გამოყენება. საინფორმაციო ტექნოლოგიების სამყაროში (IT), სერვერული ვირტუალიზაციის ცხელი თემაა. ეს ჯერ კიდევ ახალი ტექნოლოგიაა და რამდენიმე კომპანია გთავაზობთ სხვადასხვა მიდგომას.[11]

[11] https://computer.howstuffworks.com/server-virtualization.htm

თემა 7. ღრუბლოვანი გამოთვლების საფუძვლები

აქ ჩვენ განვიხილავთ ორ ძირითად ტენდენციას, რომლებიც წინ უსწრდებდნენ ღრუბლოვანი გამოთვლების კონცეფციის გამოჩენას. ეს არის IT კონსოლიდაცია და ვირტუალიზაცია. Cloud Computing მესამე ძირითადი კომპონენტი ანუ მესამე ვეშაპია კონცეფცია Software as a Service (SaaS).

მოცემული თავის მიზანია - მივიღოთ მონაცემები ღრუბლოვანი გამოთვლების წარმოშობის შესახებ, გავარკვიოთ მისი უპირატესობა და ნაკლოვანებები.

ნახ. 13. SaaS კონცეფციის გამოყენების მაგალითები.

პირველად იდეა გამოთვლების საჯარო მომსახურებაში გამოყენების შესახებ მიწოდებულ იქნა 1960-იან წლებში ინფორმაციულ ტექნოლოგიებში ცნობილი მეცნიერის *MIT*-სა და სტენფორდის უნივერსიტეტის პროფესორის ჯონ მაკარტის (John McCarthy) მიერ, მასვე ეკუთვნის ენა Lisp-ის შექმნაც. პირველი რეალური პროექტის რეალიზაცია მიეწერება კომპანია Salesforce.*com-ს,* რომელიც დაარსდა 1999 წელს. სწორედ მაშინ გაჩნდა პირველი მოსაზრება ახალი b2b სახის პროდუქტის „პროგრამული უზრუნველყოფა, როგორც მომსახურება“ ("*Software as* a *Service*", "SaaS") შესახებ. ამ სფეროში Salesforce-ის გარკეულ წარმატებას IT ინდუსტრიის გიგანტების ინტერესი მოჰყვა, რომლებმაც საჩქაროდ გამოაცხადეს საკუთარ კვლევებზე ღრუბლოვანი ტექნოლოგიის სფეროში. და აი უკვე 2005 წელს პირველი ბიზნესგადაწყვეტილება სახელწოდებით "Amazon *Web* Services" გაეშვა კომპანია Amazon.*com*-ის მიერ, რომელიც დოკტომების კრიზისის დროიდან აქტიურად ეწეოდა საკუთარი დატაცენტრების მოდერნიზაციას. შემდეგმა თავისი ტექნოლოგია

თანდათან დანერგა Google-მა, 2006 წლიდან დაიწყო b2b SaaS სერვისების მიწოდება სახელწოდებით "Google Apps".

ნახ. 14. Google-ის SaaS სერვისები.

შეგახსენებთ, რომ ღრუბლოვანი გამოთვლების ქვეშ ვგულისხმობთ პროგრამულ-აპარტულ უზრუნველყოფას, რომელიც მომხმარებლისათვის ხელმისაწვდომია ინტერნეტის ან სერვისის სახის ლოკალური ქსელის მეშვეობით. იგი მოსახერხებელი ინტერფეისის გამოყენების საშუალებას იძლევა დაშორებული რესურსებზე (გამოთვლითი რესურსები, პროგრამები და მონაცემები) ხელმისაწვდომობისათვის.

მოცემულ მომენტში ღრუბლოვანი ინფრასტრუქტურის უმეტესობა განთავსებულია დატა-ცენტრების სერვერებზე, ვირტუალიზაციის ტექნოლოგიების გამოყენებით, რაც ფაქტობრივად საშუალებას აძლევს ნებისმიერ სამომხმარებლო აპლიკაციას გამომიყენოს გამოთვლითი სიმძლავრეები ტექნიკურ ასპექტებზე დაფიქრების გარეშე. ამ შემთხვევაში „ღრუბელი“ უნდა გავიგოთ, როგორც გამოთვლებზე ერთიანი ხელმისაწვდომობა მომხმარებლის მხრიდან.

ღრუბლოვანი გამოთვლის სახეები

ღრუბლოვან გამოთვლების ცნებას ხშირად უკავშირებენ ისეთ სერვის-მომწოდებელ (Everything as a service) ტექნოლოგიებს, როგორიცაა:

- **ინფრასტრუქტურა, როგორც სერვისი“** ("Infrastructure as a Service" ანუ "IaaS")
- **„პლატფორმა, როგორც სერვისი“**("Plathform as a Service" ანუ "PaaS")
- **„პროგრამული უზრუნველყოფა, როგორც სერვისი“**("Software as a Service" ანუ "SaaS").

განვიხილოთ თითოეული ტექნოლოგია ცალ-ცალკე დეტალურად.

ინფრასტრუქტურა, როგორც სერვისი (IaaS)

IaaS - ესაა კომპიუტერული ინფრასტრუქტურის გამოყენება, როგორც სერვისი ღრუბლოვანი გამოთვლების კონცეფციის საფუძველზე.

IaaS შედგება სამი ძირითადი კომპონენტისაგან:

1. აპარატული საშუალებები (სერვერი, მონაცემთა შენახვის სისტემა, კლინეტის სისტემები, ქსელური მოწყობილობა);

2. ოპერატიული სისტემები და სისტემური პროგრამული უზრუნველყოფა (ვირტუალიზაციის საშუალებები, ავტომატიზაცია, რესურსების მართვის ძირითადი საშუალებები)

3. შუალედური პროგრამული უზრუნველყოფა (მაგალითად, სისტემის მართვისათვის).

ნახ. 15. ღრუბლოვანი ინფრასტრუქტურის კომპონენტები.

IaaS დაეფუძნებული ვირტუალიზაციის ტექნოლოგიაზე, საშუალებას აძლევს მომხმარებელს მოწყობილობა გაყოს ნაწილებად, რომელიც შეესაბამება ბიზნესის მიმდინარე მოთხოვნებს და ამით ზრდიან არსებული გამოთვლითი სიმძლავრეების ეფექტურობას.

მომხმარებელმა (კომპანია ან პროგრამული უზრუნველყოფის შემმუშავებელი) უნდა გადაიხადოს მხოლოდ მისთვის სამუშაოდ რეალურად საჭირო სერვერული დროისათვის, დისკური სივრცისა, ქსელური გამტარუნარიანობისა და სხვა რესურსებისათვის. გარდა ამისა, IaaS კლიენტს განკარგულებაში გადაეცემა მართვის ფუნქციის მთლიანი ნაკრები ერთ ინტეგრირებულ პლატფორმაში.

IaaS ათავისუფლებს საწარმოს მონაცემთა დამუშვების ცენტრის რთული ინფრასტრუქტურის, კლიენტური და ქსელური ინფრასტრუქტურის შენახვის აუცილებლობასაგან, და ასევე საშუალება აძლევს შეამციროს ამასთან დაკავშირებული კაპიტალური დანახარჯები და მიმდინარე ხარჯები. გარდა ამისა, შესაძლებელია დამატებითი ეკონომიის მიღება მომსახურების გაწევით საერთო გამოყენების ინფრასტურქტურის ჩარჩოებში.

IaaS -ის პიონერად ითვლება კომპანია Amazon, რომელიც დღეისათის ორი სახის ძირითად IaaS პროდუქტს წამოადგენენ: EC2 (**Elastic Compute Cloud**)და S3 (**Simple Storage Service**).EC2 წარმოადგენს Xen-ჰოსტინგს სტატისტიკური VPS-მახასიათებლებით, რომლებიც მყისიერად არ ფართოვდება (თუმცა ბევრი მსგავსი სერვისები უკვე წარმოდგენენ ე.წ. auto scaling-ს). S3 საცავს გააჩნია ინტერფეისი *WebDAV* და უზრუნველყოფს მუშობას სხვადასხვა ცნობილ დაპროგრამების ენასთან.

სხვა ინფრა-სერვისულ კომპანიებს შორის შეიძლება აღვნიშნოთ:

GoGrid-ს გაჩნია ძალიან მოხერხებული ინტერფეისი VPS-ს სამართავად, აგრეთვე *cloud storage* პროტოკლოების SCP, FTP, SAMBA/*CIFS*, RSYNC მხარდასაჭერად, ამასთან საცავის ზომა სკალარდება მყისიერად.

Enomaly წარმოადგენს ღრუბელში ვირტუალური აპლიკაციების გაშლისა და მართვის გადაწყვეტილებას, ამასთან მომსახურების მართვა ხორციელდება ბრაუზერის მეშვეობით. სასიამოვნო დამატებას წარმოადგენს ვირტუალური მანქანების ავტომატური მასშტაბირება მიმდინარე დატვირთვისას, ასევე დატვირთვის ავტობალანსირება. მხარდჭერილი ვირტუალური არქიტექტურებს შორის არის Linux, Windows, Solaris და BSD Guests. ვირტუალიზაციისათვის გამოიყენება არა მარტო Xen არამედ *KVM და* VMware-ც კი.

Eucalyptus წარმოადგენს პროგრამულ კომპლექსს ღია კოდით *cloud computing* -ის რეალიზაციისათვის კლასტერულ სისტემაზე.

პლატფორმა როგორც სერვისი (PaaS)

PaaS - ეს არის ინტეგრირებული პლატფორმების გამოყენება ვებ-აპლიაკაციების, როგორც სერვსის: შემუშავების, ტესტირების, გაშლისა და მხარდაჭერისათვის.

ვებ-აპლიკაციების გაშლისათვის არ არის აუცილებელი მოწყობილობებისა და პროგრამული უზრუნველყოფის შეძენა, არ არის მათი მხარდამჭერი ორგანიზების არსებობის აუცილებლობა. კლიენტის ხელმისაწვდომობის ორგანიზება შესაძლებელია არენდის პირობებში.

ასეთ მიდგომას გააჩნია შემდეგი ღირსებები:

- სკალაცია;
- გაუმართაობისადმი ტოლერანტობა;
- ვირტუალიზაცია;
- უსაფრთხოება.

სკალაცია *PaaS* გვთვაზობს საჭირო რესურსების ავტომატურ გამოყოფას და გათავისუფლებას მომხმარებელთა მომსახურე აპლიკაციების რაოდენობაზე დამოკიდებულებით.

PaaS, როგორც ინტეგრირებული პლატფორმა შემუშვების, ტესტირების, გაშლისა და ვებ-აპლიკაციების მხარდაჭერისათვის საშუალებას იძლევა მთელი რიგი შემუშვება, ტესტირება და ვებ-აპლიკაციების გაშლასთან დაკავშირებული ოპერაციები შესრულდეს ინტეგრირებულ სფეროში და ამით გამორიცხავს სხვა ცალკეული ეტაპებისთვის საჭირო სფეროებზე დამატებით დანახარჯებს.

საწყისი კოდის შექმნის უნარი და შემუშავების ბრძანების შიგნით მისი საერთო ხელმისაწვდომობა, არსებითად ზრდის აპლიკაციის *PaaS* ბაზაზე შექმნის მწარმოებლურობას. ასეთი პლატფორმის ყველაზე ცნობილი მაგალითია AppEngineGoogle-ისაგან, რომელიც გვთავაზობს ჰოსტინგს ვებ-აპლიკაციებისათვის, დამატებითი გამომთვლელი რესურსების შეძენის შესაძლებლობით (მაგალითად, მაღალი დატვირთვის ტესტირებისათვის). Google AppEngine აპლიკაციის ვირტუალურ კლასტერულ სისტემებზე

გასაშვებად შემუშავდა პლატფორმა AppScale, რომელსაც მაინც არ ჰქონდა არანაირი კავშირი Google-თან.

ვებ-ძებნის სისტემებში და კომპანიის კონტექსტურ რეკლამაში Yahoo-მ გამოიყენა პლატფორმა Hadoop, რომელიც ორიენტირებულია დიდი მოცულობის მონაცემების გადაცემაზე ქსელურ სერვერებს შორის. Hadoop-ის ბაზაზე აიგო Hbase(Google BigTables-ის მონაცემთა ბაზის ანალოგი), ასევე HDFS(Hadoop Distributed File System, аналог Google File System).

PaaS –ის კიდევ ერთი მკაფიო წარმომადგენელია კომპანია Mosso-ს პროდუქტი :

- Cloud Sites - ვებ-ჰოსტინგი(Linux, Windows, Mail) დამტვირთველი ვებ-პროექტებისათვის შესაძლებლობით გააფართოვონ ბაზური უფასო შესაძლებლობები დამატებითი გადახდებით (ტრაფიკი, მონაცემთა საცავები, გამოთვლითი სიმძლავრე).
- Cloud Files - ფაილური cloud-ჰოსტინგი შესანახ ფაილების მოცულობაზე თითოეულ გიგაბაიტზე ყოველთვიური გადახდებით. მართვა ხორციელდება ბრაუზერის მეშვეობოთ, ან API-ს საშუალებით (PHP, Python, Java, .NET, Ruby).
- Cloud Servers - სერვერის საფოსტო არენა (Ram საათში), სერვერული ოპერაციული სისტემის არჩევის შესაძლებლობით. ხოლო ღრუბლოვანი ინფრასტრუქტურის ცენტრშია Microsoft - ოპერაციული სისტემა Windows Azure. Windows Azure ქმნის ერთიან გარემოს, რომელიც მოიცავს ღრუბლოვან ანალოგებს Microsoft-ის სერვერული პროდუქტების (SQL Azure რელაციური მონაცემთა ბაზა, გვევლინება SQL Server-ის, აგრეთვე Exchange Online, SharePoint Online და Microsoft Dynamics CRM Online) და შემუშავების ინსტრუმენტების ანალოგად (.NET Framework და Visual Studio2010 წლის ვერსია, აღჭურვილი ნაკრებით Windows Azure Tools). ასე მაგალითად, პროგრამისტი, რომელიც ქმნის საიტს Visual Studio 2010-ში, შეუძლია აპლიკაციიდან გამოუსვლელად განათავსოს თავისი გვერდი Windows Azure-ში.

პროგრამული უზრუნველყოფა როგორც სერვისი (SaaS)

SaaS - აპლიკაციის გაშლის მოდელი, რომელიც გულისხმობს საბოლოო მომხმარებლებისათვის აპლიკაციის მიწოდებას, როგორც მომსახურება მოთხოვნით (on demand). შეღწევადობა ამ აპლიკაციაში ხორციელდება ქსელის მეშვეობით, ხშირად მხოლოდ ინტრენეტ-ბრაუზერის საშუალებით. მოცემულ შემთხვევაში SaaS მოდელის კლიენტისათვის

ძირითადი უპირატესობაა დანახარჯების არ არსებობა, რომელიც დაკავშირებულია მასზე მომუშავე მოწყობილობის მუშაობისუნარიანობასათან და პროგრამული უზრუნველყოფის ჩატვირთვა, განახლებასა და მხარდაჭერასათან. მისი მიზნობრივი აუდიტორია საბოლოო მომხმარებელია.

ნახ.16. ღრუბლოვანი ტექნოლოგიების აისბერგის მწვერვალი წარმოდგენილი SaaS სერვისებით.

მოდელში SaaS:

- მოწყობილობა ადაპტირებულია დისტანციურ გამოყენებაზე;
- ერთი აპლიკაციით სარგებლობა შეუძლია რამოდენიმე კლიენტს;
- გადასახადი მომსახურებაზე ამოიღება, როგორც ყოველთვიური სააბონენტო გადასახადი, ან ტრანზაქციის ჯამური მოცულობის საფუძველზე;
- აპლიკაციის მხარდაჭერა შედის გადახდილ თანხაში;
- აპლიკაციის მოდერნიზაცია შეიძლება განხორციელდეს პერსონალის მიერ კლიენტისათვის შეუფერხებლად და გამჭირვალედ.

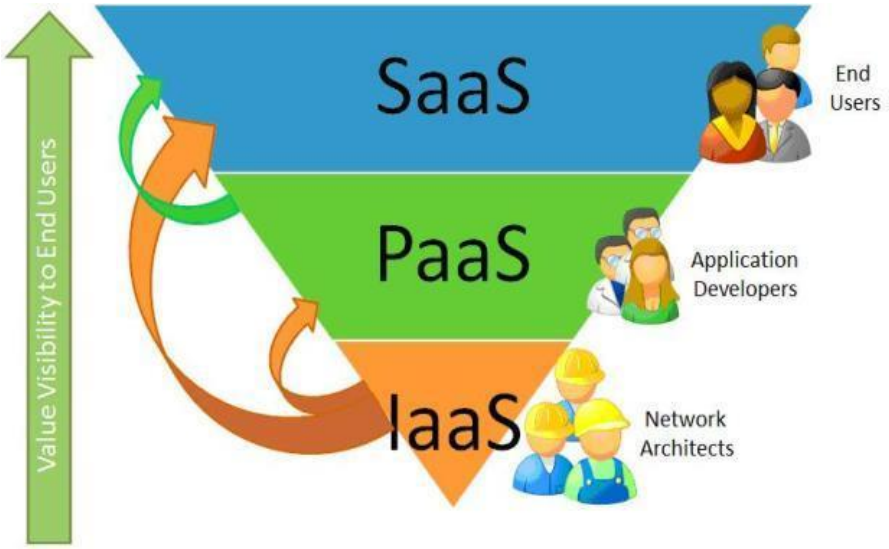
პროგრამული უზრუნველყოფის შემმუშავებლების თვალსაზრისით, SaaS მოდელი პროგრამული უზრუნველყოფის არალიცენზირებული გამოყენებასთან ეფექტურად ბრძოლის საშუალებას იძლევა, იმის წყალობით, რომ კლიენტს არ შეუძლია შეინახოს, დააკოპიროს და დააყენოს პროგრამული უზრუნველყოფა.

თავისი არსით, პროგრამული უზრუნველყოფა SaaS-ის ჩარჩოებში შეიძლება განვიხილოთ, როგორც შიდა ინფორმაციული სისტემების უფრო მოსახერხებელი და მომგებიანი ალტერნატივა.

SaaS ლოგიკის განვითარების კონცეფციას *WaaS (Workplace* as a Service-სამუშაო ადგილი, როგორც მომსახურება) წარმოადგნეს. კლიენტი იღებს განკარგულებაში სამუშაოდ საჭირო მთლიანად აღჭურვილ პროგრამული უზრუნველყოფის ვირტუალურ სამუშაო ადგილს.

SoftCloud მოთხოვნით სარგებლობს SaaS-ის შემდეგი აპლიკაციები (პოპულარობის შემცირების რიგითობით):

- ფოსტა;
- კომუნიკაცია (VoIP)
- ანტისპამი და ანტივირუსი;
- Helpdesk
- პროექტების მართვა;
- დისტანციური სწავლება;
- CRM
- მონაცემების შენახვა და დარეზერვება



ნახ.17. SaaS სერვისებს გააჩნიათ ყველაზე დიდი სამომხმარებლო ბაზა.

მსგავსი პროდუქტებია: MobileMe (Apple), Azure (Microsoft) და LotusLive (IBM). მოცემული მომსახურების არსი იმაში მდგომარეობს, რომ ისინი მომხმრებელს აძლევენ საკუთარი მონაცემების (კონტაქტები, ფოსტა, ფაილები) შენახვაზე წვდომას, ასევე რამოდენიმე მომხმარებლის დოკუმენტებზე ერთობლივი მუშაობის საშუალებას.

მომხმარებელთა მონაცემების ინტერნეტში შენახვის საკითხებმა კომპანია Google-იც შეაწუხა, რომელმაც შეიმუშავა პროექტი GDrive, რომელიც იქნება წარმოდგენილი მყარი დისკის სახით და განისაზღვრება ოპერაციული სისტემის მიერ როგორც ლოკალური. ასევე გაცხადებულია, რომ შესაძლებელი იქნება უზარმაზარი რაოდენობის მონცემის შენახვა, რაც მიმზიდველად გამოიყურება.

შეუზღუდავად ფაილების შენახვას გვთავაზობს ასევე MediaFire.com. არსებობს მისი, როგორც სრულიად უფასო გამოყენების შესაძლებლობა (მართალია ზოგიერთი შეზღუდვით, მაგალითად როგორიცაა ჩასატვირთი ფაილის ზომა), ასევე ხდება პრემიუმ-აკაუნტის ყიდვა, რომელიც აფართოებს შესაძლებლობებს (მაგალითად, ფაილის დაშიფვრა, პირდაპირი გზავნილების გადმოქაჩვაზე წვდომის მიღება).

SaaS-ის კიდევ ერთი საინტერესო წარმომადგენელია პროდუქტი iCloud. იგი წარმოადგენს ოპერაციულ სისტემას, რომლითაც მუშაობა შესაძლებელია უშუალოდ ბრაუზერის მეშვეობით. ოპერაციული სისტემის ინტერფეისი შესრულებულია Windows Vista/XP-ის სტილში. ამჟამად პროექტი ბეტა სტადიაზე იმყოფება და თვითონ ოპერაციულ სისტემაში აპლიკაციების მინიმუმია რეალიზებული.

SaaS-ს მიეკუთვნება ასევე *Online backup* მომსახურებები, ანუ, მარტივად, რომ ვთქვათ - მონაცემთა სარეზერვო კოპირება. მომხმარებელი უბრალოდ იხდის სააბონენტო გადასახადს, ხოლო სერვისები ავტომატურად, განსაზღვრულ დროს ახდენენ მონაცემთა დაშიფვრას კომპიუტერიდან ან სხვა მოწყობილობიდან და აგზავნიან დაშორებულ სერვერს, ამგვარად მონაცემები ხელმისაწვდომი ხდება დედამიწის ნებისმიერი წერტილიდან. ასეთ მომსახურებას ახლა მრავალი კომპანია გვთავაზობს, მათ შორისაა Nero და Symantec.

cloud-ტექნოლოგიების საინტერესო გამოყენება მოძებნეს კომპიუტერული თამაშების შექმნელებმა: ახლა თანამედროვე კომპიუტერებს და სათამაშო კონსოლებს აღარ დასჭირდებათ მძლავრი გრაფიკული ადაპტერები (ვიდეობარათები), რადგან მონაცემთა მთლიანი დამუშავება და რენდერინგი მოხდება cloud -სერვერების მიერ, ხოლო მოთამაშეები მიიღებენ უკვე დამუშავებულ ვიდეოს.

SaaS-კონცეფციის თანახმად მომხმარებელი პროდუქტის ყიდვისას მაშინათვე არ იხდის არამედ იღებს მას კრედიტით. ამასთან იყენებს იმ ფუნქციებს რაც მას სჭირდება. მაგალითად, თუ თქვენ წელიწადში ერთხელ გჭირდებათ რომელიღაც პროგრამა და უფრო ხშირად არ

აპირებთ მის ხმარებას, მაშინ რატომ უნდა იყიდოთ პროდუქტი რომელსაც არ გამოიყენებთ? რატომ უნდა დააკავებინოთ ადგილი (ბინაში, თუ ეს დისკით ყუთია, და ვინჩესტერზე თუ ეს ფაილია)?

ღრუბლოვან სფეროში კონკურენციამ მიგვიყვანა უფასო სერვისების გაჩენამდე. სწორედ ამ გზით წავიდა ორი კონკურენტი - Microsoft და Google. ორივე კომპანიამ გამოუშვა მომსახურებების ნაკრები, რომლებიც დუკუმენტებთან მუშაობის საშუალებას იძლევა. Google- ეს Google Docs, ხოლო Microsoft-სთან - Office Web Apps.

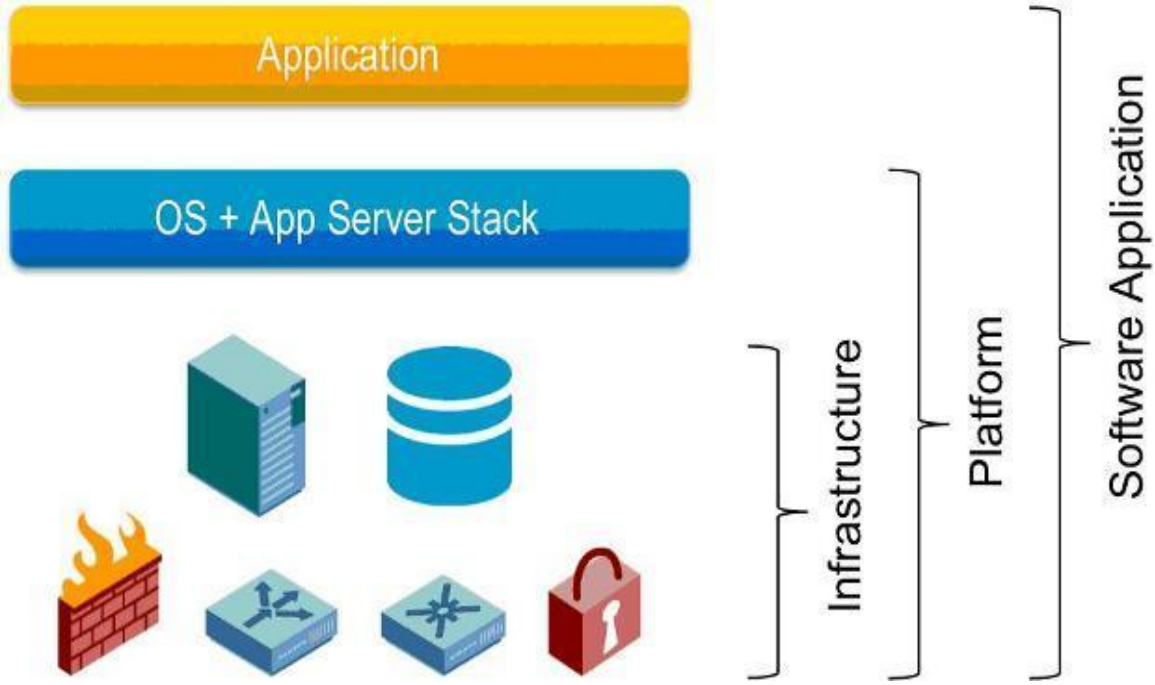
ამასთან ორივე სერვისი მჭიდროდ დაკავშირებულია ფოსტასთან (Gmail -თან პირველ შემთხვევეში და Hotmail-თან მეორე შემთხვევაში) და ფაილების საცავებთან. ამგვარად, მომხმარებელი თითქოსდა გადაჰყავთ, მისთვის ჩვეული ოფლაინ-გარემოდან ონლაინში. მნიშვნელოვანია, რომ Google-იც და Microsoft-იც თავიანთი ონლაინ-სერვისების მხარდაჭერის ინტეგირებას ახდენენ ყველა პროგრამულ-გარემოში - როგორც სამაგიდო, ისე მობილურში (შეგახსენებთ, რომ Google-მა შექმნა ОС *Android*, ხოლო Microsoft -მა - Windows Phone 8).

ანალოგიურ კონცეფციას (მაგრამ ცოტა სხვა აქცენტებით) წინ სწევს ორივე კომპანიის მთავარი კონკურენტიც - Apple. საუბარი მიდის ძალიან საინტერესო სერვისზე სახელწოდებით MobileMe. სერვისი თვის თავში მოიცავს საფოსტო კლიენტს, კალენდარს, მისამართების წიგნს, საფაილო საცავს, სურათების ალბომს და დაკარგული iPhone-ის მოსაძებნად ინსტრუმენტს. ამ მომსახურების გამოსაყენებლად Apple იღებს დაახლოებით 65 ევროს (ან 100 დოლარს) წელიწადში. ამასთან Apple უზრუნველყოფს ინტერნეტ-სევისების და აპლიკაციების საკუთარი ნაკრებების ისეთი დონის ურთიერთმოქმდებას კომპიუტერზე (Mac OS X-ის მართვის ქვეშ), ტელეფონზე, ფლეერზე და iPad-ზე, რომ ბრაუზერის გამოყენება საჭირო აღარაა. თქვენს Mac, iPhone და iPad-ზე ხმარობთ თქვენთვის ჩვეულ პროგრამებს, მაგრამ ყველა მონაცემი ინახება არა მათზე, არამედ ღრუბელში, რაც საშუალებას გაძლევთ დაივიწყოთ სინქრონიზაციის აუცილებლობის შესახებ, ასევე - მათ ხელმისაწვდომობაზე.

თუ Apple აერთიანებს ვებ-სერვისებს ოპერაციული სისტემის ჩვეულ აპლიკაციებში, Google საწინააღმდეგო მხრიდან შედის: ინტერნეტ გიგანტის მიერ შემუშავებული ოპერაციული სისტემა Chrome OS წარმოადგენს, ფაქტობრივად, ერთი ბრაუზერს, რომლის მეშვეობითაც მომხმარებელი ურთიერთობს ვებ-მომსახურების დატოტვილ ქსელთან. OS ორიენტირებულია

ნეთბუქებზე, გამოირჩევა ძალიან დაბალი სისტემური მოთხოვნით და პროგრამების დამოუკიდებელი ინსტალაციის საჭიროების არ არსებობით (რადგან ყველა პროგრამა უშუალოდ მუშაობს ვებში). ე.ი. Google ჩვეულებრივ კლიენტებს სთავაზობს ღრუბლოვან კონცეფციებს, რომლებიც ჩვეულებრივ დემარკირებულები არიან კორპორატიულ კლიენტებთან მუშაობისას. ამავე დროს, აშკარაა, რომ ქსელური მაუწყებლობის ინტერნეტის არასაკმარისი ფართოზოლოვანი შეღწევის მქონე ქვეყნებში ასეთი ნეთბუქების გამოყენება შეუძლებელია. იმიტომ რომ Chrome OS-ის ბაზაზე ნეთბუქი ინტერნეტის გარეშე სრულიად გამოუსადეგი იქნება.

სამივე ტიპის ღრუბლოვანი მომსახურება ურთიერთდაკავშირებულია და წარმოადგენს წყობრ სტრუქტურას.



ნახ. 18. ღრუბლოვანი სერვისების ურთიერთკავშირი

მომსახურების გაწევის სხვადასხვა გზების გარდა განასხვავებენ ღრუბლოვანი სისტემების განლაგების რამდენიმე ვარიანტს:

კერძო ღრუბელი (private cloud) - გამოიყენება ერთი კომპანიის შინით მომსახურების გაწევისათვის, რომელიც ერთდროულად მომსახურების შემკვეთსაც და მიმწოდებელსაც წარმოადგენს. „ღრუბლოვანი კონცეფციის“ რეალიზების ეს მაგალითი, როდესაც კომპანია ორგანიზაციის ჩარჩოებში ქმნის მას საკუთარი თავისთვის. *private cloud*-ის რეალიზაცია

პირველ რიგში ხსნის ერთ-ერთ საკითხს, რომელიც აუცილებლად უჩნდებათ შემკვეთებს, რომლებიც ამ კონცეფციას ეცნობიან - მონაცემთა დაცულობის შესახებ ინფორმაციული უსაფრთხოების თვალსაზრისით. რამდენადაც „ღრუბელი“ შემოსაზღვრულია თვითონ კომპანიის ჩარჩოებში, ამდენად ეს საკითხი წყდება არსებული სტანდარტული მეთოდებით. *private cloud* -ისთვის დამახასიათებელია მოწყობილობების ღირებულების დაწევა გაცდენილი ან არაეფექტურად გამოყენებული რესურსების ხარჯზე. ასევე მოწყობილობების შეძენაზე გაწეული ხარჯის დაწევა ლოგისტიკის შეკვეცის ხარჯზე (არ ვფიქრობთ რომელი სერვერები შევიძინოთ, როგორ კონფიგურაციაში, რომელი საწარმოო სიმძლავრეები, რა რაოდენობის ადგილი დავარეზერვოთ ყოველ ჯერზე და ა.შ.).

არსებითად, სიმძლავრე იზრდება ზოგადად მზარდი დატვირთვის პროპორციულად, თითოეული წარმოშობილი ამოცანისაგან დამოუკიდებლად - არამედ, ასე ვთქვათ, საშუალოდ და მარტივდება დაგეგვმაც, ყიდვაც და რეალიზებაც - ხდება ახალი ამოცანების გაშვება წარმოებაში.

საჯარო ღრუბელი - გამოიყენება ღრუბლოვანი პროვაიდერების მიერ გარეშე მომხმარებელისთვის მომსახურების გასაწევად.

შერული (ჰიბრიდული) ღრუბელი - აღნიშნული ორი განლაგების მოდელების ერთობლივი გამოყენება.

ზოგადად Cloud-ის ერთ-ერთ ძირითად იდეა მდგომარეობს სწორედ იმაში, რომ ტექნოლოგიური თვალსაზრისით შიგა და გარეშე ღრუბლებს შორის განსხვავება არ იყოს და შემკვეთს შეეძლოს მოქნილად გადაადგილოს თავისი დავალებები თავის და გაქირავებულ IT-ინფრასტუქტურას შორის იმაზე ფიქრის გარეშე თუ სად სრულდება ისინი კონკრეტულად.

ნახ. 19. სხვადასხვა ტიპის ღრუბლებს შორის ურთიერთკავშირი.

ამდენად, ამ ტექნოლოგიების ერთობლივი გამოიყენება, ღრუბლოვანი გამოთვლების მომხმარებლებს საშუალებას მისცემს გამოიყენონ კომპიუტერული სიმძლავრის და მონაცემთა შენახვის შესაძლებლობა, რაც უშუალოდ გარკვეულ ვირტუალიზაციის ტექნოლოგიების მეშვეობით და აბსტრაქციის მაღალი დონით წარედგინებათ როგორც მომსახურება.

ღრუბლოვანი გამოთვლების ღირსებები

განვიხილოთ ღრუბლოვანი გამოთვლების ძირითადი უპირატესობანი და ღირსებები:

ხელმისაწვდომობა და გაუმართაობის მიმართ ტოლერანტობა - ყველა მომხმარებლისათვის ხელმისაწვდომობა ნებისმიერი წეტილიდან სადაც არის ინტერნეტი, ნებისმიერი კომპიუტერიდან, სადაც არის ბრაუზერი.

კლიენტების კომპიუტერები. მომხმარებლებისათვის არაა აუცილებელი შეიძინონ ძვირადღირებული კომპიუტერი, დიდი მოცულობის მეხსიერებით და დისკით იმისათვის, რომ გამოიყენონ პროგრამები ვებ-ინტერფეისით. გარდა ამისა, CD და DVD დისკები არ არის საჭირო, რადგან ყველა ინფორმაცია და პროგრამა "ღრუბელში" რჩება. მომხმარებელს შეუძლიათ ჩვეულებრივი კომპიუტერებისა და ლეპტოპებიდან უფრო კომპაქტურ და მოსახერხებელ ნეთბუქებზე გადასვლა.

დოკუმენტებთან წვდომა. თუ დოკუმენტები ინახება „ღრუბელში“, მომხმარებლისათვის ისინი ხელმისაწვდომი იქნებიან ნებისმიერ დროს და ნებისმიერ ადგილას. უკვე აღარ არსებობს ისეთი ცნება როგორიცაა დავიწყებული ფაილები: თუ არის ინტერნეტი ისინი მუდამ გვერდითაა.

მონაცემების დაკარგვესთან ან მოწყობილობების ქურდობასთან მდგრადობა. თუ მონაცემები ინახება „ღრუბელში“, მათი ასლები ავტომატურად ვრცელდება რამდენიმე სერვერით, რომელებიც შესაძლებლია სხვადასხვა კონტინენტზე მდებერეობდნენ. პერსონალური კომპიუტერის ძარცვის ან გატეხვის შემთხვევაში მომხმარებელი არ კარგავს ღირებულ ინფორმაციას, რომელიც მას შეუძლია მიიღოს ნებისმიერი სხვა კომპიუტერიდან.

საიმედოობა. დატა ცენტრებს მართავენ პროფესიონალი სპეციალისტები, რომლებიც უზრუნველყოფენ ვირტუალური მანქანების სადღეღამისო ფუნქციონირებას. თუნდაც ფიზიკური მანქანა „გაფუჭდეს“, აპლიკაციების მრავალ ასლად განაწილების ხარჯზე ის მაინც

განაგრძობს თავის სამუშაოს. ეს ქმნის საიმედოობის განსაზღვრულად მაღალ დონეს და სისტემის ფუნქციონირების გაუმართაობის მიმართ ტოლერანტობას.

ეკონომიურობა და ეფექტურობა - გადაიხადე იმდენი რამდენსაც მოიხმარ, თავს უფლება მიეცი გამოიყენო ძვირფასი, მძლავრი კომპიუტერები და პროგრამები. „ღრუბელი“ საშუალებას გაძლევთ განიხილოთ და გადაიხადოთ მხოლოდ რეალურად მოხმარებული რესურსებისათვის მკაცრად, მათი გამოყენების ფაქტის მიხედვით;

რესურსების დაქირავება. ჩვეულებრივ საშუალო კომპანიის სერვერები ჩატვირთულია 10-15%-ით. გარკვეული პერიოდის განმავლობაში არსებობს დამატებითი კომპიუტერული რესურსების საჭიროება, სხვებში კი ეს ძვირადღირებული რესურსები უმოქმედოა. ნებისმიერ დროს "ღრუბელში" საჭირო რაოდენობის გამოთვლითი რესურსების გამოყენებით კომპანიები ამცირებენ დანახარჯს მოწყობილობებზე და მათ მომსახურებაზე. ეს მომხმარებელს საშუალებას აძლევს უარი თქვას ძვირადღირებული IT-აქტივების შეძენაზე, არა ქირაობის სასარგებლოდ არამედ საჭიროებიდან გამომდინარე ოპერატიული მოხმარების, თავისი სისტემების მომსახურებაზე დანახარჯის შემცირების და მიმწოდებლისაგან სერვისის დონის გარანტიის მიღების პირობებში.

პროგრამული უზრუნველყოფის დაქირავება. თითოეული ლოკალური მომხმარებლისათვის პროგრამული პაკეტების ყიდვის მაგივრად, კომპანია ყიდულობს საჭირო პროგრამებს „ღრუბელში“. მოცემული პროგრამები გამოყენებული იქნება მხოლოდ იმ მომხმარებლების მიერ, რომლებისთვისაც ეს პროგრამები აუცილებელია მუშაობაში. გარდა ამისა, ინტერნეტის საშუალებით ხელმისაწვდომობაზე ორიენტრებული პროგრამების ღირებულება მნიშვნელოვნად დაბალია პერსონლური კომპიუტერებისთვის მათ ანალოგებთან შედარებით. თუ პროგრამა ხშირად არ გამოიყენება, მაშინ მათი დაქირავება შეიძლება საათობრივი ანაზღაურებით. პროგრამის განახლებაზე და ყველა სამუშაო ადგილზე მუშა მდგომარეობაში მათ მხარდაჭრაზე დანახარჯები ნულამდეა დაყვანილი.

IT-მომსახურებების მომწოდებელთათვის ღრუბლის ეკონომიკური არსი *მასშტაბის ეფექტურობაში* (დიდი ერთგვაროვანი ცენტრის მომსახურება უფრო იაფია, ვიდრე სხვადასხვაგვარი უამრავი პატარის) და დატვირთვის შერბილებაში (როდესაც მომხმარებელი ბევრია, გამორიცხულია, რომ პიკური სიმძლავრეები ყველა მათგანს ერთდროულად მოუნდეს) მდგომარეობს.

პროგრამული უზრუნველყოფის შემმუშავებლები ასევე იღებენ სარგებელს ღრუბელზე გადასვლით: ახლა მათთვის მარტივია, სწრაფად და იაფად შეიმუშავონ, დატვირთვის დროს მოახდინონ ტესტირება და კლიენტებს შესთავაზონ თავიანთი გადაწყვეტილებები - ეს პირდაპირ შეიძლება გაკეთდეს ღრუბელში მინიმალური ხარჯით. გარდა ამისა, ღრუბლოვანი გამოთვლები - ეს არის SaaS-ის ფორმაში პროგრამული უზრუნველყოფის დამოუკიდებელი მწარმოებლებისათვის მოგების და გაყიდვების არხის გაზრდის ეფექტური ინსტრუმენტი. ეს მიდგომა მომსახურების დინამიური წარმოდგენის ორგნიზების საშუალებას იძლევა, როდესაც მომხმარებლებს შეუძლიათ გადაიხადონ ფაქტების მიხედვით და მოახდინონ თავიანთი რესურსების რეგულირება რეალური მოთხოვნილებაზე დამოკიდებულებით ვადამდელი ვალდებულებების გარეშე.

სიმარტივე - არ არის საჭირო პროგრამების და მოწყობილობების ყიდვა და კონფიგურაცია, მათი განახლება.

მომსახურება. რამდენადაც ფიზიკური სერვერები *Cloud Computing*-ის დანერგვასთან ერთად პატარავდება, მათი მომსახურება მარტივი და სწრაფი ხდება. რაც შეეხება პროგრამულ უზრუნველყოფას, უკანასკნელი დამონტაჟებული, დაკონფიგურირებული და განახლებულია "ღრუბელში". ნებისმიერ დროს როდესაც მომხმარებელი უშვებს დაშორებულ პროგრამას, ის დარწმუნებული უნდა იყოს, რომ ეს პროგრამის უკანასკნელი ვერსიაა - რაიმე გადანაცვლების ან განახლებისათვის გადახდის საჭიროების გარეშე.

თანამშრომლობა. "ღრუბელში" დოკუმენტებთან მუშაობისას არ არის საჭირო ერთმანეთისათვის ვერსიების გაგზავნა ან შემდგომში მათი რედაქტირება. ახლა მომხმარებელს შეუძლია დარწმუნებული იყოს, რომ მათ წინაშე დოკუმენტის უახლესი ვერსიაა და ერთი მომხმარებლის მიერ გაკეთებული ნებისმიერი ცვლილება მყისიერად აისახება მეორეში.

ღია ინტერფეისები. "ღრუბელს" ჩვეულებრივ აქვს სტანდარტული ღია API-ები (გამოყენებითი პროგრამირების ინტერფეისები) არსებული აპლიკაციების კომუნიკაციისთვის და ახალი პროგრამების - სპეციალურად ღრუბლოვან არქიტექტურაზე - შექმნისათვის.

მოქნილობა და სკალარულობა - გამოთვლითი რესურსების შეუზღუდავობა (მეხსიერება, პროცესორი, დისკი). „ღრუბელი“ სკალარულია და ელასტიური - რესურსები გამოიყოფა და თავისუფლდება საჭიროების შესაბამისად;

პროდუქტიული გამოთვლები. პერსონალური კომპიუტერებთან შედარებით გამოთვლითი, რომელიც ხელმისაწვდომია „ღრუბლოვანი“ კომპიუტერების მომხმარებლისთვის, პრაქტიკულად შემოსაზღვრულია მხოლოდ „ღრუბლის“ ზომით, ე.ი. დაშორებული სერვერების საერთო როდენობით. მომხმარებლებს შეუძლიათ გაუშვან უფრო რთული ამოცანები, აუცილებლი მეხსიერების დიდი რაოდენობით, მონაცემთა შენახვის ადგილით, როცა ეს აუცილებელია. სხვა სიტყვებით, სურვილის მიხედვით მომხმარებელს შეუძლია ადვილად და იაფად იმუშაოს სუპერკომპიუტერით ფაქტობრივი შეძენის გარეშე. მრავალ ვირტუალურ მანქანაზე აპლიკაციების უამრავი ასლის გაშვების შესაძლებლობა სკალარულობას უპირატესობას ანიჭებს: აპლიკაციების ეგზემპლიარების რაოდენობა პრაქტიკულად მომენტალურად იზრდება მოთხოვნის მიხედვით, დატვირთვიდან გამომდიმნარე.

მონაცემთა შენახვა. პერსონალური კომპიუტერებზე ინფორმაციის შესანახ ხელმისაწვდომ ადგილთან შედარებით „ღრუბელში“ საცავის მოცულობას შეუძლია მოქნილად და ავტომატურად აეწყოს მომხმარებლის საჭიროებიდან გამომდინარე. „ღრუბელში“ ინფორმაციის შენახვისას მომხმარებლებს შუძლიათ დაივიწყონ იმ შეზღუდვების შესახებ რომელსაც ადებენ ჩვეულებრივი დისკები. „ღრუბლოვანი“ ზომები განისაზღვრება მილიარდობით გიგაბაიტის ხალმისაწვდომი ადგილით.

სტარტაპებისათვის ინსტრუმენტი. Cloud Computing-ის ისეთი მომხმარებლების თვალში, როგორებიც არიან კომპანიები, როლებიც ბიზნესს ახალად იწყებენ, ამ ტექნოლოგიების უპირატესობა ისაა, რომ არ არის საჭირო ყველა აუცილებელი აპარატურისა და პროგრამული უზრუნველყოფის შესყიდვა და შემდეგ მათი მუშაობის ხელშეწყობა.

ღრუბლოვანი გამოთვლების ნაკლი და პრობლემები

ქსელთან მუდმივი კავშირი. Cloud Computing ყოველთვის მოითხოვს ქსელთან მუდმივ კავშირს (ინტერნეტს). თუ არაა ქსელში წვდომა - არაა სამუშაო, პროგრამები, დოკუმენტები. მრავალი „ღრუბლოვნი“ პროგრამა მოითხოვს კარგ ინტერნეტ-შეერთებას, მაღალი გამავლობის უნარით. შესაბამისად პროგრამებმა შეიძლება უფრო ნელა იმუშაონ ვიდრე ლოკალურ კომპიუტერებზე.

უსაფრთხოება

მონაცემთა დაცულობა თეორიულად შეიძლება საფრთხის ქვეშ აღმოჩნდეს. ყველა მონაცემი არ შეიძლება ანდო ინტერნეტში მესამე მხარის პროვაიდერს, მითუმეტეს არა მარტო შენახვისათვის, არამედ დამუშავებისთვისაც. ყველაფერი დამოკიდებულია იმაზე თუ ვინ უზრუნველყოფს „ღრუბლოვან" სერვისებს. თუ ვინმე უსაფრთხოდ ინახავს თქვენს მონაცემებს, მუდმივად იღებს მათ სარეზერვო ასლებს, მუშაობს მრავალი წლის განმავლობაში მსგავსი სერვისების ბაზარზე და კარგი რეპუტაცია აქვს, მაშინ მონაცემთა დაცულობას საფრთხე არ ემუქრება. „ღრუბელოვანი" ბიზნეს აპლიკაციების მომხმარებელს ასევე შეიძლება ჰქონდეს სამართლებრივი პრობლემები, მაგალითად, პერსონალური მონაცემების დაცვის მოთხოვნების შესრულებასთან დაკავშირებით.

სახელმწიფოს, რომლის ტერიტორიაზეც განთავსებულია მონაცემთა ცენტრი, მასში შენახული ნებისმიერი ინფორმაციის მიღება შეუძლია. მაგალითად, აშშ-ში სადაც მდებარეობს ყველაზე დიდი რაოდენობის დატაცენტრები, კანონონმდებლობის თანახმად, პროვაიდერ-კომპანიას არ აქვს უფლება, მისი ადვოკატების გარდა, გაახმიანოს კონფიდენციალური ინფორმაციის გადაცემის ფაქტი.

ეს პრობლემა, ალბათ, ერთ-ერთი ყველაზე მნიშვნელოვანია ღრუბელში კონფიდენციალური ინფორმაციის გაცემის საკითხში. ამის გადაჭრის რამდენიმე გზა არსებობს. პირველი, შეიძლება დაიშიფროს მთელი ინფორმაცია, რომელიც ღრუბელზეა მოთავსებული. მეორე, ის უბრალოდ არ უნდა განალაგო მასში. მაგრამ, ყოველ შემთხვევაში, კომპანიას, რომელიც სარგებლობს ღრუბლოვანი გამოთვლებით, ეს საკითხი უნდა ჰქონდეს ინფორმაციული უსაფრთხოების სიაში განსაზღვრულ პუნქტად. გარდა ამისა, თვითონ პროვაიდერებმა უნდა გააუმჯობესონ თავიათი ტექნოლოგიები, წარმოადგინონ რა დაშიფვრაში ზოგიერთი მომსახურება.

„ღრუბელოვანი“ აპლიკაციების ფუნქციონალობა. ყველა პროგრამა ან მათი თვისებები არაა ხელმისაწვდომი დისტანციურად. თუ შევადარებთ ადგილობრივი გამოყენების პროგრამებს მათ "ღრუბელოვან" ანალოგებს, ეს უკანასკნელი კვლავ ჩამორჩებიან ფუნქციონირებაში. მაგალითად, Google Docs ცხრილებს ან Office web application აპლიკაციებს გაცილებით ნაკლები ფუნქცია და შესაძლებები აქვთ, ვიდრე Microsoft Excel-ს.

„ღრუბლოვან“ პროვაიდერებზე დამოკიდებულება. ყოველთვის რჩება რისკი, რომ ონლაინური სერვისების პროვაიდერი ერთხელ არ გააკეთებს მონაცემების სარეზერვო ასლს - სწორედ სერვერის მწყობრიდან გამოსვლის წინ. ეს რისკი ალბათ არ აღემატება იმ საფრთხეს, რომ მომხმარებელი თვითონ დათმობს თავის მონცემებს - დაკარგავით ან მობილურის ან ნოუთბუქის დაზიანებით, საშინაო პერსონალურ კომპიუტერზე სარეზერვო ასლის შეუქმნელად. გარადა ამისა, ვეჩვევით რა ამათუიმ მომსახურებას, ჩვენ რაღაც ხარისხით ვზღუდავთ საკუთარ თავისუფლებას - ძველ ვერსიაზე გადასვლის, ინფორმაციის დამუშავების ხერხის არჩევის და ა.შ. თავისუფლებას.

ზოგიერთი ექსპერტი, მაგალითად ჰ. მაკლოედი (Hugh Macleod) სტატიაში „ღრუბლების ყველაზე კარგად დაცული საიდუმლო“, ამტკიცებს, რომ ღრუბლოვან გამოთვლებს მივყავართ უზარმაზარ, ადრე არნახულ მონოპოლიის შექმნამდე. ეს ნამდვილად შესაძლებელია? რა თქმა უნდა, ღრუბლოვანი გამოთვლების ბაზარზე ღრუბელში რაიმე ინფორმაციის განთავსებისათვის, რომლის მიმართაც არსებობს ინფორმაციული უსაფრთხოების წესი, კომპანია ალბათ ისეთ ვენდორებს გამოიყენებს, რომელთა სახელები „ცნობადია“ და რომლებსაც ენდობია. ამგვარად, არსებობენ განსაზღვრული საფრთხეები იმისა, რომ ყველა გამოთვლა და მონაცემი აგრერირებული იქნება ერთი ზემონოპოლიის ხელში. მაგრამ ამჟამად ბაზარზე უკვე არსებობს რამდენიმე კომპანია კლიენტების მხრიდან სანდოობის დაახლოებით ერთნაირად მაღალი დონით (Microsoft, Google, Amazon), და არ არსებობობს არანაირი ფაქტი, რომლიც მიუთითებდა ერთი კომპანიის დანარჩენებზე დომინირებას. ამიტომ უახლოეს მომავალში გლობალური ზეკომპანიის გამოჩენა, რომელიც მოახდენს კოორდინირებას და გააკონტროლებს მთელ გამომთვლელ სამყაროს, ნაკლებად სავარაუდოა, მაგრამ ამის შესაძლებლობაც კი აფრთხობს ზოგიერთ კლიენტს.

რა არის ღრუბლოვანი გამოთვლები?

ზუსტად რომ განვმარტოთ, ღრუბლოვანი გამოთვლები - ეს არის ისეთი გამოთვლითი მომსახურებების წარმოდგენა, როგორიცაა სერვერები, საცავები და სხვა, ინტერნეტის მეშვეობით. კომპანიებს, რომლებიც გვთავაზობენ ამ გამოთვლითი სერვისებს, ეწოდებათ Cloud მომწოდებლები. ისინი იღებენ გადასახადს Cloud Computing სერვისების გამოყენების მიხედვით.

ღრუბლოვანი გამოთვლები ჩვეულებრივ კლასიფიცირდება ადგილმდებარეობის, ან იმ სერვისზე საფუძველზე, რომელსაც Cloud სთავაზობს.

ღრუბლის ადგილმდებარეობაზე დაყრდნობით, ჩვენ შეგვიძლია მოვახდინოთ ღრუბლის კვალიფიცირება როგორც:

- საჯარო,
- პირადი,
- ჰიბრიდული
- სათემო ღრუბელი.

სერვისის საფუძველზე, რომელსაც ღრუბელი გვაზობს, ჩვენ მას მივაკუთვნებთ:

- IaaS (ინფრასტრუქტურა, როგორც სერვისი)
- PaaS (პლატფორმა-როგორც-სამსახური)
- SaaS (პროგრამული უზრუნველყოფა, როგორც სერვისი)
- ან, როგორც მონაცემთა ბაზას, ინფორმაციას, პროცესს, აპლიკაციას, ინტეგრაციას, უსაფრთხოებს, მენეჯმენტს, ტესტირებას როგორც სერვისს.

მონაცემების ტიპზე დამოკიდებულებით, რომლითაც თქვენ მუშაობთ, თქვენ მოგინდებათ რომ შეადაროთ საჯარო, კერძო და ჰიბრიდული ღრუბლები სხვადასხვა დონის უსაფრთხოებისა და მენეჯმენტის მოთხოვნებს.

- საზოგადოებრივი Cloud - მთლიანი კომპიუტერული ინფრასტრუქტურა მდებარეობს იმ ღრუბლოვანი კომპიუტერული კომპანიის შენობაში, რომელიც გთავაზობთ ღრუბლოვან მომსახურებას.
- პირადი Cloud - ყველა თქვენი კომპიუტერული ინფრასტრუქტურის ჰოსტინგი საკუთარია და არავისთან არ არის გაზიარებული. უსაფრთხოებისა და კონტროლის დონე ყველაზე მაღალია კერძო ქსელის გამოყენებისას.
- ჰიბრიდული ღრუბელი - იყენებს როგორც კერძო, ასევე საზოგადოებრივ ღრუბლებს, მათი დანიშნულების მიხედვით. თქვენ ათავსებთ თქვენს ყველაზე მნიშვნელოვან აპლიკაციებს საკუთარ სერვერებზე, რათა შეინარჩუნოთ ისინი უფრო უსაფრთხოდ, ხოლო მეორეხარისხოვანს - სხვაგან.
- სათემო ღრუბელი - საზოგადოებრივ ღრუბელს იყენებენ ერთობლოვად ორგანიზაციები, რომლებსაც აქვთ საერთო მიზნები, ან რომლებიც შეესაბამება

კონკრეტულ თემს (საზოგადოებას) (პროფესიულ საზოგადოებას, გეოგრაფიულ საზოგადოებას და ა.შ.).

ღრუბლოვანი სერვისების სახეები: IaaS, PaaS, SaaS, FaaS

ღრუბლოვანი მომსახურება 4 კატეგორიად იყოფა: ინფრასტრუქტურა როგორც სერვისი (IaaS), პლატფორმა, როგორც სერვისი (PaaS), პროგრამული უზრუნველყოფა, როგორც სერვისი (SaaS) და FaaS (ფუნქციები, მომსახურება). მათ ზოგჯერ უწოდებენ Cloud Computing სტელაჟს, რადგან ისინი ერთმანეთის თავზე იგება.

1. ინფრასტრუქტურა, როგორც სერვისი (IaaS)

IaaS არის Cloud Computing სერვისების უმნიშვნელოვანესი კატეგორია, რომელიც საშუალებას გაძლევთ იქირავოთ IT ინფრასტრუქტურა (სერვერები ან VM- ს) ღრუბლოვანი პროვაიდერისგან ანაზღაურების საფუძველზე.

2. პლატფორმა, როგორც სერვისი (PaaS)

პლატფორმა როგორც-სერვისი (PaaS)-ის დანუშნულებაა ვებ-გვერდის ან მობილური აპლიკაციების სწრაფად შექმნისათვის ინფრასტრუქტურული სერვერის, საცავის, ქსელის მონაცემთა ბაზების სწრაფი შექმნა.

3. პროგრამული უზრუნველყოფა, როგორც სერვისი (SaaS)

პროგრამული უზრუნველყოფა, როგორც სერვისი (SaaS) წარმოადგენს ინტერნეტის საშუალებით პროგრამული უზრუნველყოფის პროგრამების მიწოდების მეთოდს, მოთხოვნის და სააბონენტოს საფუძველზე.

4. FaaS (ფუნქციები, როგორც მომსახურების)

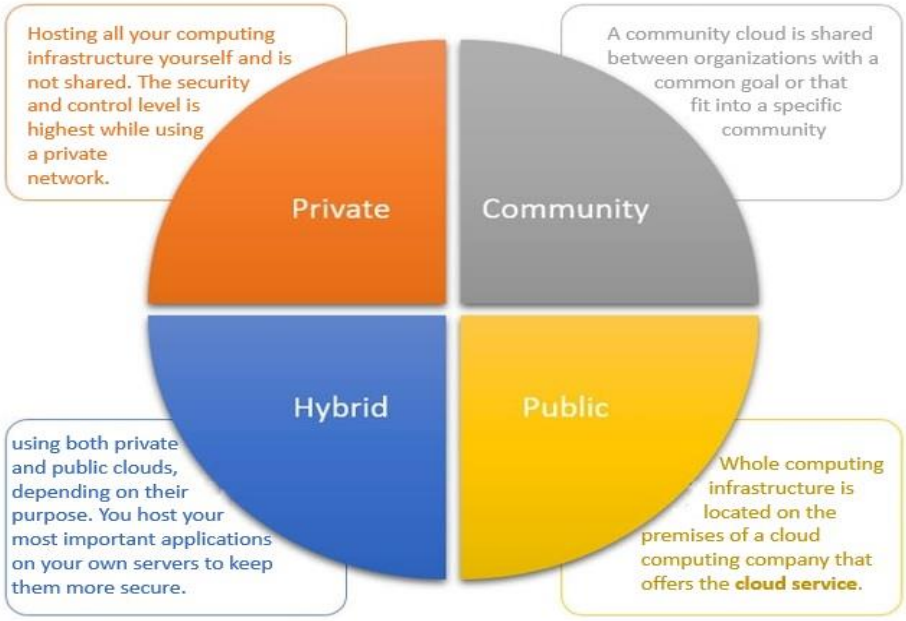
FaaS ამატებს კიდევ ერთი დონის აბსტრაქციას PaaS, ისე, რომ მიმწოდებლები მთლიანად იზოლირებულები არიან ყველაფერ იმისაგან, რაც სტელაჟში მათი კოდის ქვეშ მდებარეობს. ვირტუალური სერვერების, კონტეინერების და აპლიკაციების runtimes- ის hassles- ის ნაცვლად, ისინი ტვირთავენ კოდების ვიწრო ფუნქციურ ბლოკებს განსაზღვრავენ მათ გაშვებას განსაზღვრული მოვლენის მოხდენის დროს, FaaS აპლიკაციები არ მოიხმარენ IaaS რესურსებს სანამ მოვლენა არ ხდება, რაც ამცირებს სარგებლობის საფასურს.

ღრუბლოვანი გამოთვლების გამოყენება

თუმცა ალბათ თქვენ ვერ აცნობიერებთ იმას, რომ სწორედ ახლა Cloud Computing- ს იყენებთ, უმეტესობა ჩვენგანი ვიყენებთ ონლაინ-მომსახურებას ელექტრული ფოსტის გასაგზავნად, დოკუმენტების რედაქტირებისათვის, ფილმების საყურებლად და სხვა. სავარაუდოა, რომ Cloud Computing ამას ყველაფერს სცენარის მიღმა აკეთებს. დღეს სხვადასხვა ორგანიზაციები, რომლებიც მცირე ზომის სტარტაპებიდან სამთავრობო უწყებებამდე იყენებენ ამ ტექნოლოგიას შემდეგი მიზნებისათვის:

- ახალი აპლიკაციების და სერვისების შექმნა, აგრეთვე მონაცემების შენახვა, დარეზერვება და კოპირება.
- ჰოსტინგ საიტები და ბლოგები.
- აუდიო და ვიდეოს გავრცელება
- მოთხოვნის პროგრამული უზრუნველყოფის სერვისი
- მონაცემების ანალიზი
- პროგნოზირება.[12]

Cloud Computing - Cloud- ის სახეები



[12] https://www.esds.co.in/blog/cloud-computing-types-cloud/#sthash.5mhDt8JU.ajz4STqh.dpbs

რა არის ღრუბლოვანი გამოთვლები

Cloud Computing არის ინფორმაციული ტექნოლოგიების (IT) მომსახურების მიწოდების მეთოდი, რომლის საშუალებითაც ვებ-საშუალებებით და აპლიკაციებით ხდება რესურსების ამოღება ინტერნეტიდან, და არა სერვერთან პირდაპირ კავშირით. იმის ნაცვლად, რომ ფაილები შეინახული იყოს მყარ დისკზე ან ლოკალურ საცავ მოწყობილობაზე, ღრუბელოვანი საცავი საშუალებას იძლევა ისინი შეინახული იყოს დისტანციური მონაცემთა ბაზაში. სანამ ელექტრონულ ხელსაწყოს აქვს წვდომა ქსელთან, მას აქვს წვდომა მონაცემებთან და მათი გაშვებისთვის პროგრამებთან.

ამას ღრუბლოვანი გამოთვლები ეწოდება, რადგან ინფორმაცია რომელზედაც ხდება ხელმისაწვდომობა "ღრუბელში" არის ნაპოვნი და არ საჭიროებს იმას, რომ მომხმარებლი იყოს კონკრეტული ადგილას მასთან მისაწვდომობის მიზნით. ამ ტიპის სისტემა საშუალებას აძლევს თანამშრომლებს იმუშაონ დისტანციურად. Cloud Services- ის კომპანიები საშუალებას აძლევენ მომხმარებლებს შეინახონ ფაილები და აპლიკაციები დისტანციური სერვერების მეშვეობით და შემდეგ მიიღონ მათთან წვდომა ინტერნეტის საშუალებით.

ღრუბლოვანი გამოთვლების ჩაშლა(BREAKING DOWN)

თავისი არსით, ღრუბლოვანი გამოთვლების იდეა არის ის, რომ ყველა მძიმე სამუშაო, რომელიც დაკავშირებულია მონაცემთა დამონტაჟებისა და დამუშავების პროცესთან, გადატანილი იქნას იმ მოწყობილობიდან, რომელსაც ატარებთ თან ან მუშაობთ მასზე, დიდ კომპიუტერულ კლასტერებზე შორეულ კიბერსივრცეში. ინტერნეტი ხდება ღრუბელი და voilà - თქვენი მონაცემები, სამუშაოები და აპლიკაციები ხელმისაწვდომია ნებისმიერ აპარატში, რომელთანაც შეგიძლიათ ინტერნეტთან დაკავშირება, მსოფლიოს ნებისმიერ წერტილში.

ბიზნეს მენეჯმენტის კონსულტანტი ფირმის Forrester-ის მიერ ჩატარებული კვლევის თანახმად, Cloud Computing- ის ბაზარი **BREAKING DOWN** 2020 წლისთვის 191 მილიარდ დოლარს მიაღწევს.

Cloud Computing- ის უპირატესობები

Cloud-ზე დაფუძნებული პროგრამული უზრუნველყოფის ზრდამ ყველა სექტორის კომპანიას რამდენიმე შეღავათი შესთავაზა, მათ შორის, ნებისმიერი მოწყობილობიდან პროგრამული უზრუნველყოფის გამოყენების უნარი, ან საკუთარი აპლიკაციით ან ბრაუზერის მეშვეობით. შედეგად, მომხმარებლებს აქვთ საშუალება განახორციელონ თავიანთი ფაილების და პარამეტრების სხვა მოწყობილობებზე გადატანა სრულიად დაუბრკოლებლად. Cloud Computing ბევრად უფრო მეტია, ვიდრე მხოლოდ რამდენიმე მოწყობილობაში ფაილებისადმი წვდომა. Cloud-computing სერვისების წყალობით მომხმარებლებს შეუძლიათ შეამოწმონ თავიანთი ელ.ფოსტა ნებისმიერ კომპიუტერზე და შეინახონ ფაილები, როგორიცაა Dropbox და Google Drive. Cloud Computing მომსახურება ასევე საშუალებას აძლევს მომხმარებლებს შექმნან თავიანთი მუსიკის, ფაილების და ფოტოების სარეზერვო ასლები, რაც ახდენს იმის უზრუნველყოფას, რომ ფაილები დაუყოვნებლივ ხელმისაწვდომი იყოს მყარი დისკის გაუმართაობის შემთხვევაში.

Cloud Computing გთავაზობთ მსხვილ ბიზნესს სერიოზული ხარჯების დაზოგვის პოტენციალს. სანამ Cloud გახდებოდა ეფექტური ალტერნატივა, კომპანიები იძულებული იყვნენ, შეეძინათ, შეექმნათ და შეენარჩუნებინათ ინფორმაცის მართვის ძვირადღირებული ტექნოლოგიები და ინფრასტრუქტურა. ახლა, იმის მაგივრად, რომ ჩაიდოს მილიონობი მსხვილი სერვერების ცენტრების და გლობალური IT განყოფილებების ინვესტირებაში, რომელიც მუდმივად განახლებას მოითხოვს, ფირმას შეუძლია გამოიყენოს ავტომატიზირებული სამუშაო ადგილის "სინათლის" ვერსიები, ელვისებური ინტერნეტ-კავშირებით, ხოლო თანამშრომლები ერთმანეთს დაუკავშირდებიან Cloud ონლაინში, რათა შექმნან პრეზენტაციები, ელ.ცხრილები და კომპანიის პროგრამული უზრუნველყოფა.

ადამიანები აღმოაჩენენ, რომ როდესაც ისინი ჩატვირთავენ ფოტოებს, დოკუმენტებსა და ვიდეოებს ღრუბლში და შემდეგ იღებენ იქიდან თავიანთი შეხედულებისამებრ, ეს უზრუინველყოფს მათ სამაგიდო კომპიუტერების ან ლეპტოპების მეხსირებაში ადგილის ეკონომიას. გარდა ამისა, ღრუბლოვანი სტრუქტურა მომხმარებლებს საშუალებას აძლევს უფრო სწრაფად განაახლონ პროგრამული უზრუნველყოფები - იმიტომ, რომ პროგრამული უზრუნველყოფის შემმუშავებელ კომპანიებს შეუძლიათ თავიანთი პროდუქციის შეთავაზება ინტერნეტის საშუალებით, და არა ტრადიციული, ხელშესახები მეთოდების გამოყენებით, რომელიც იყენებს დისკებს ან ფლეშ დრაივებს. 2013 წელს, Adobe Systems-მა გამოაცხადა Adobe

Photoshop-ის შემდგომი ვერსიები, ისევე როგორც მისი შემოქმედებითი Suite-ის სხვა კომპონენტები, რომლებიც ხელმისაწვდომია მხოლოდ ინტერნეტზე დაფუძნებული ხელმოწერის საშუალებით. ეს საშუალებას აძლევს მომხმარებლებს გადმოიწერონ ახალი ვერსიები და ადვილად შეასწორონ თავიანთი პროგრამები.

Cloud Computing- ს ნაკლოვანებები

Cloud Computing-ის სიჩქარეს, ეფექტურობასა და ინოვაციასთან ერთად ჩნდება რისკები.

თავდაპირველად, უსაფრთხოება განიხილებოდა როგორც არასასურველი ღრუბლის გამოყენების დროს, განსაკუთრებით მაშინ, როდესაც საუბარი მიდიოდა მგრძნობიარე სამედიცინო ჩანაწერებსა და ფინანსურ ინფორმაციის შესახებ. თუმცა ნორმატიული მოთხოვნები აიძულებენ Cloud Computing- ის სამსახურებს, უზრუნველყონ თავიანთი უსაფრთხოებისა და შესაბამისობის ზომები, ის კვლავაც მუდმივად პრობლემური რჩება. მედიის სათაურები მუდმივად ყვირიან ამა თუ იმ კომპანიის მონაცემთა დარღვევების შესახებ, რომლის დროსაც მგრძნობიარე ინფორმაცია მოხვდა მავნე ჰაკერების ხელში, რომლებსაც შეუძლიათ წაშალონ, მანიპულირება მოახდინონ ან სხვაგვარად გამოიყენონ მონაცემები (თუმცა, ზოგიერთი ანგარიშის თანახმად, მონაცემთა გატეხვების უმეტესობა დაკავშირებული იყო ლოკალურ და არა ღრუბლოვან სისტემებთან). Encryption იცავს სასიცოცხლო ინფორმაციას, მაგრამ თუ დაშიფვრის გასაღები დაკარგულია, მონაცემები ქრება.

Cloud computing კომპანიების მიერ დაცული სერვერები შეიძლება დაზარალდნენ სტიქიური უბედურებით, შიდა შეცდომებით და ენერგიის გათიშვაზეც. სამწუხაროდ, Cloud Computing- ის გეოგრაფიული მოცვა გამოთვლებს ორივე მიმართულებით წყვეტს: კალიფორნიაში ელექტროენერგიის გათიშვამ შეიძლება მოახდინოს ნიუ-იორკში მომხმარებლების პარალიზება; ტეხასში ფირმამ შეიძლება დაკარგოს მონაცემები თუ ვიღაც იწვევს Maine-ში (შტატი ჩრდილოეთ ამერიკაში) დაფუძნებული მისი პროვაიდერის შეფერხებას.

საბოლოო ჯამში, როგორც ნებისმიერ ახალი ტექნოლოგიაში, თანამშრომლებისა და მენეჯერებისათვის არსებობს სწავლების მრუდი. მაგრამ ბევრ ადამიანს აქვს ინფორმაციაზე ხელმისაწვდომობა და მანიპულირებენ ერთი პორტალით, უნებლიე შეცდომები შეიძლება მთელს სისტემაში გადავიდეს.

რა ახდენს ღრუბლოვანი გამოთვლების ზრდის სტიმულირებას?

Cloud Computing- ის ერთ-ერთ ყველაზე დიდი დაბრკოლებას ინტერნეტ სიჩქარე წარმოადგენს: ჩვენ გვჭირდება, რომ ინტერნეტი იყოს სუპერ სწრაფი, ჩქარი ნაკადი, ისევე სწრაფად მოძრაობდს, როგორც სადენში სადმე სახლებში ან ოფისში. საბოლოოდ ჩვენ ამას მივაღწიეთ ფართოზოლოვანი მიღების ფართო გავრცელების ხარჯზე და 3G და 4G უკაბელო ტექნოლოგიის გამოყენებით. ჩვენ ასევე გვიწევს დაველოდოთ ვიდრე ინტერნეტ უსაფრთხოების სტანდარტები და პროტოკოლები საკმარისად მყარი გახდება იმისათვის, რომ მთავარ აღმასრულებელ ხელმძღვანელებს უსაფრთხოდ შეეძლოთ მონაცემთა კლასტერების გატანა თავიანთი შენობებიდან და სხვის ხელში გადაცემა.

მაგრამ ახლა, რომ მათ აქვთ და აცნობიერებენ დანაზოგების პოტენციალს, რომელიც დაკავშირებულია ტექნოლოგიური სერვისებისთვის საჭირო პროგრამებისა და ტექნიკური უზრუნველყოფის გარეწყაროდან შექმნის უნართან, სიჩქარემ, რომლის საშუალებითაც კომპანიები ინტერნეტის საშუალებით მოქმედებენ და ინტერნეტ-სისტემებით სარგებლობენ, მოიმატა. Nasdaq- ის თანახმად, ინვესტიციებმა ძირითადი სტრატეგიული მიმართულებებით, როგორიცაა დიდი მონაცემთა ანალიზი, კორპორატიული მობილური კავშირი, უსაფრთხოება და ღრუბლოვანი ტექნოლოგია, 2018 წლისთვის 40 მილიონ დოლარზე მეტით გაიზარდა.

ბიზნეს Cloud Computing-ის სამყარო

კომპანიებს შეუძლიათ გამოიყენონ Cloud Computing სხვადასხვანაირად. ზოგიერთი მომხმარებელი შეინარჩუნებს ყველა აპლიკაციას და მონაცემს Cloud- ზე, მაშინ როცა სხვები იყენებენ ჰიბრიდულ მოდელს - გარკვეულ აპლიკაციებს და მონაცემებს კერძო სერვერებზე ინახავენ, ხოლო ზოგიერთს ღრუბელში.

რაც შეეხება მომსახურების გაწევას, კორპორატიული გამოთვლების სფეროში დიდ მოთამაშეებს წარმოადგენენ:

- Google Cloud
- Amazon Web Services
- Microsoft Azure
- IBM Bluemix
- Aliyun

Amazon Web Services (AWS) არის 100%-ით საჯარო და მოიცავს ანგარიშების დროულად გადახდის მოდელს და არა გარე მოდელს. ერთ პლატფორმაზე ყოფნისას თქვენ შეგიძლიათ დარეგისტრირდეთ აპლიკაციებსა და დამატებით სერვისებზე. Google Cloud, რომელიც გამიზნულია სამომხმარებლო ბანკინგისა და საცალო ვაჭრობისთვის, არის ერთ-ერთი ბოლო მონაწილე. Microsoft Azure, რომელმაც ცოტა ხნის წინ აამუშავა მონაცემთა ცენტრები დიდ ბრიტანეთში, საშუალებას აძლევს კლიენტებს შეინახოს გარკვეული მონაცემები საკუთარ საიტებზე.

ინვესტიციების ვარიანტები ღრუბლოვან გამოთვლებში

ღრუბელზე დაფუძნებულ მომსახურებებთან ერთად, მოსალოდნელია, რომ მომავალში ღრუბლოვანი მომსახურებები გეომეტრიული პროგრესიით გაიზრდება, ინვესტიციებისათვის არასდროს ყოფილა უკეთესი დრო, მაგრამ მნიშვნელოვანია ამის ფრთხილად გაკეთება. (იხ. პრაიმერი ტექნოლოგიების ინდუსტრიაში ინვესტიციების შესახებ). Cloud-ზე დაფუძნებული საინვესტიციო ვარიანტების არჩევისას, გახსოვდეთ, რომ არსებობს სექტორში ჩართული მრავალი განსხვავებული ელემენტი, რომელთაგან თითოეული გვთავაზობს სხვადასხვა შესაძლებლობას. მცირე კომპანიები, რომლებიც მხოლოდ ღრუბლოვან გამოთვლებზეა ორიენტირებული, უფრო ძვირია იმასთან შედარებით რამდენსაც გამოიმუშავებენ დღეს. ამგვარი კომპანიები უფრო სარისკოები არიან, მაგრამ თუ მართლაც ხდება ღრუბლოვანი გამოთვლების ფლობა - და ყველა ნიშანი მიუთითებს მის ფართოდ გავრცელებაზე - ეს უფრო ფოკუსირებული თამაშები მსხვილ კომპანიებს აიძულებს სიფრთხილით დაიწყონ საქმე. მაგრამ არ უნდა უგულებელვყოთ ის პოტენციური უპირატესობა, რომლის ნახვაც შეეძლო ისეთ უზარმაზარი კომპანიებსაც კი, როგორცაა IBM ან Microsoft. იმის მიხედვით, მსხვილი კორპორაციები როგორ იწყებენ რაოდენობის შემცირებას და მათი IT განყოფილებების აუთსორსინგის ნაწილების პოტენციურ დაზოგვას, ზოგიერთი მსხვილი შეკვეთა შეიძლება დახვდეთ ტექნიკური სექტორის გიგანტებს გზაზე.

მართლაც, აღიარებული კომპიუტერული კომპანიები, როგორიცაა აშშ პროგრამული გიგანტური Oracle Corporation (NYSE: ORCL), დაშორდნენ ტრადიციულ პროგრამულ უზრუნველყოფას და განახორციელეს ინვესტიციები ღრუბლოვან გამოთვლებში. Oracle- მა 2015 წლის მეოთხე კვარტალში შეიძინა 3600 კლიენტი და ღრუბლოვანი გამოთვლების ბიზნესიდან მისმა შემოსავალმა 690 მილიონ დოლარს მიაღწია. როგორც Cantor Fitzgerald- ის

ხუთვარსკვლავიანი ანალიტიკოსი ბრაიან უაიტი აღნიშნავს, Oracle-ის გადასვლა Cloud- ზე "როგორც ჩანს, უფრო სწრაფად ხდება, ვიდრე კომპანია მოელოდა."

ერთი გაფრთხილება: რამდენედაც ღრუბლოვანი გამოთვლები ახლა ძალიან პოპულარულია, მრავალი ფირმა მიისწრაფის მასში მონაწილეობა მიიღოს. დახარჯეთ დრო, რათა გულდასმით განიხილონ ზუსტად რა არის ის, რასაც კომპანია გთავაზობთ, და უზრუნველყოფს, ისინი უბრალოდ ხომ არ იყენებენ დარგობრივ ჟარგონს, რომ გამოიყენონ საბაზარო ინტერესი.

აპარატურული თამაშები

კომპანიები, როგორიცაა Google (NASDAQ: GOOG), IBM (NYSE: IBM), Intel (NASDAQ: INTC), Microsoft (Nasdaq: MSFT), Cisco (Nasdaq: CSCO) და Hewlett-Packard (NYSE: HPQ) დებენ ღრუბლოვან გამოთვლებში უზარმაზარ ინვესტიციებს. ისინი ქმნიან სერვერების ზღვებს, რომლებიც ღრუბლოვანი გამოთვლების საშუალებას იძლევიან. ზოგიერთებს, როგორცაა Google და Microsoft აქვთ ინტერნეტის საშუალებით შემოთავაზებული საკუთარი აპლიკაციები, ხოლო IBM და HP ფირმები უფრო მეტად დაინტერესებული არიან მსხვილი კორპორატიული მომხმარებლებისთვის საყრდენით უზრუნველყოფით.

მრავალი კომპანია ასევე აშენებს ცენტრალიზებულ მონაცემთა ცენტრებს. ზოგიერთი მათგანი პიონერია ინტერნეტ-პროვაიდერების ინდუსტრიაში, როგორიცაა, მაგალითად, Rackspace, რომელიც ფლობს Apollo Global Management (NYSE: APO); სხვები, როგორიცაა Amazon.com (Nasdaq: AMZN), ვებ- სამყაროს სხვა სფეროებდან მოდიან. მიუხედავად იმისა, რომ Amazon მომხმარებელთათვის ცნობილია, როგორც ინტერნეტ სუპერმაღაზია, ის ასევე წარმოადგენს საბაზრო ლიდერს Cloud Computing სექტორში. უკანასკნელ პერიოდში, კომპანია განაგრძობს მილიარდობით დოლარის ინვესტირებას, აფართოებს რა მის AWS მონაცემთა გადამუშავების ცენტრებს ქვეყნის მასშტაბით და მთელს მსოფლიოში და მუშაობს ახალ ღრუბლოვან მომსახურებაზე, რომელიც ორიენტირებულია ხელოვნური ინტელექტის დარგზე, რომელსაც ეწოდება ღრმა სწავლება ან მანქანური სწავლება (ML), იგი ხელს უწყობს კომპიუტერების „გაწვრთნას“ საუბრის, სურათებისა და ობიექტების ამოცნობაში.

ასევე არსებობს უამრავი პატარა კომპანია, რომლებიც მუშაობენ ღრუბლოვანი გამოთვლებისათვის ინტერნეტისა და კორპორატიული IT ცენტრის განახლებაზე. Akamai (Nasdaq: AKAM) მონდომებით მუშაობს იმაზე, რომ ინტერნეტის "მილებს" შეეძლოთ

უზარმაზარი რაოდენობით მონაცემების გადაქაჩვა, რომელიც საჭიროა ღრუბლოვანი გამოთვლების რეალობად საქცევად.

პროგრამული თამაშები

ეს ტენდენცია არ ეხება არა მხოლოდ მოწყობილობებს. ასევე უნდა მოხდეს პროგრამული უზრუნველყოფის შეცვლა იმისათვის, რომ ღრუბლოვანმა გამოთვლებმა იმუშაონ. თქვენს კომპიუტერში პროგრამების ინსტალაციის ან უზარმაზარი IT-სტაფის , საკუთარი სერვერების ფერმებში განახლების ნაცვლად, პროგრამული აპლიკაციების ექსკლუზიურად მიწოდება და დაინსტალირება მოხდება ინტერნეტის მეშვეობით. როგორც ზემოთ იყო აღნიშნული, პროგრამული უზრუნველყოფა, როგორც სერვისი სწრაფად იზრდება. ინოვაციური კომპანიები, როგორიცაა Salesforce.com (NYSE: CRM) და Concur Technologies (Nasdaq: CNQR), იღებენ ისეთ პოპულარული აპლიკაციებს, როგორიცაა ხარჯების შესახებ ანგარიშგება, სამოგზაურო ლოგისტიკა და კონტაქტების მართვა, და მათ ეს შეეთავაზათ როგორც SaaS. მიუხედავად იმისა, რომ ზოგიერთი ყველაზე ცნობილი ბრენდები შეძენილი და გადაცემული იყო კერძო საკუთრებაში (RIP Keynote Systems) ზოგჯერ თქვნ შეგიძლიათ ინვესტირება განახორციელიოთ მათი მშობელ კომპანიაში. საჩვენებელი მაგალითი: ადამიანური რესურსების მართვის პროგრამული უზრუნველყოფა Taleo, ახლა ეკუთვნის Oracle-ს.

გამყიდველები, რომლებიც სპეციალიზდებიან Cloud-ზე დაფუძნებულ ფაილების გაცვლასა და შენახვაზე, წარმოადგენენ სხვა განსხვავებულ ვარიანტს. Apple (Nasdaq: AAPL) და Google ჩართული არიან თამაშში, როგორც Dropbox და Mozy, რომელიც ეკუთვნის EMC Corporation-ს (NYSE: EMC).

უსაფრთხოება რჩება სასიცოცხლოდ მნიშვნელოვან პრობლემად, ინტერნეტში ონლაინ ფაილების წვდომის დროს. სულ უფრო და უფრო მეტი კომპანია წარმოადგენს ღრუბლოვანი უსაფრთხოების უზრუნველყოფს გადაწყვეტილებებს, მათ შორის Forcepoint (თანადაფინანსების მფლობელი Raytheon (NYSE: RTN) და Qualys (NASDAQ: QYLS).

Cloud-ზე დაფუძნებული გადაწყვეტილებები ხშირად გვთავაზობენ ზოგიერთი ტიპის კომპიუტერულ ვირტუალიზაციას ან აპლიკაციის ტექნოლოგიას. ვირტუალიზაციის ტექნოლოგიის წამყვანი მომწოდებლებს მიეკუთვნებიან Citrix (Nasdaq: CTXS) და VMware (NYSE: VMW).[13]

[13] https://www.investopedia.com/terms/c/cloud-computing.asp

თემა 8.ღრუბელში ვებ-სერვისები

ვებ სერვისები და ღრუბლოვანი გამოთვლები

ამ თავის მიზანს წარმოადგენს იმ ვებ-სერვისების მიმოხილვა, რომელიც წარმოდგენილია ღრუბლოვანი გამოთვლების კონცეფციით. განსაკუთრებული ყურადღება ექცევა „ინფრასტრუქტურა როგორც სერვისი“ ტიპს.

რამდენადაც ტექნოლოგიები მიგრირებენ ტრადიციული ლოკალური მოდელიდან ახალ ღრუბლის მოდელში, სერვერული შეთავაზებები პრაქტიკუად ყოველდღიურად ვითარდება. ვებ-მომსახურებების შეთავაზებებს ხშირად გააჩნიათ საერთო მახასიათებლები. ხშირად კლიენტისაგან მოითხოვება მხოლოდ მინიმალური *დანახარჯი* მომსახურების მისაღებად. სკალარულობა გათვალისწინებულია შემოთავაზების თითოეული ტიპისათვის, ეს ყოველთვის აუცილებელი არაა. მრავალი „ღრუბლოვანი“ ვენდერებიდან კიდევ მუშობენ სკალარულობის გამოყენებაზე, რადგან მათი მომხმარებლები ჯერ არ საჭიროებენ ამ სახის მომსახურებას. ბოლოს, მოწყობილობები და ადგილმდებარეობაზე დამოუკიდებლობა საშუალებას აძლევს მომხმარებელს მიიღონ სისტემებთან წვდომა იმისგან დამოუკიდებლად, თუ სად იმყოფებიან ან რომელ მოწყობილობებს იყენებენ.

ინფრასტუქტურა როგორც სერვისი (IaaS)

ინფრასტრუქტურა როგორც სერვისი (*Infrastructure-as-a-Service, IaaS)* - ეს არის კომპიუტერული ინფრასტრუქტურის (როგორც წესი - ვირტუალიზაციის პლატფორმა), მომსახურების სახით წარმოჩენა. IaaS მნიშვნელოვნად აძლიერებს ტექნოლოგიას, მოსახურებებს და დაბანდებების მონაცემთა დამუშავებას ცენტრებში, იმისათვის რომ შეთავაზებული იყოს ეს როგორც სერვისი კლიენტებისადმი. ტრადიციული აუტსორსინგისაგან განსხვავებით, რომელიც სათანადო ძალისხმევას, მოლაპარაკებებს და რთულ, გრძელ კონტრაქტებს მოითხოვს, IaaS ორიენტირებულია მომსახურების მიწოდების მოდელზე, რომელიც უზრუნველყოფს წინასწარ განსაზღვრულ, სტანდარტირეზებულ ინფრასტრუქტურას, აუცილებლად ოპტიმიზირებულს მომხმარებლის საჭიროებების მიხედვით. სამუშაოს შეთავაზების გამარტივება და მომსახურების სერვისის დონის არჩევა უადვილებს შემდეგ კლიენტს ძირითდი ექსპლუატაციური მახასიათებლების განსაზღვრული

ნაკრებით გადაწყვეტილების არჩევას. როგორც წესი, მომწოდებლები გვათავაზობენ შემდეგი დონეების კომპონენეტებს:

- აპარატული უზრუნველყოფა (როგორც წესი, ეს არის გრიდი მასიური ჰორიზონტალური სკალარულობით);
- კომპიუტერული ქსელი (მოიცავს მარშრუტიზატორებს, ბრანდმაუერებს, დატვირთვის ბალანსირებას და ა.შ.);
- ინტერნეტთან მიერთება;
- ვირტუალიზაციის პლატფორმა იმისათვის, რომ გაშვებული იყოს ვირტუალურული მანქანები;
- სერვისულ მომსახურებაზე თანხმობა;
- საბუღალტრო აღრიცხვის ინსტრუმენტები.

მონაცემთა ცენტრების, სერვერების, პროგრამული უზრუნველყოფის, ქსელური აღჭურვილობისა და ა.შ. სივრცის შესყიდვის ნაცვლად, IaaS მომხმარებელი იჯარით იღებს რესურსებს, რომლებიც განლაგებულია მომსახურების IaaS პროვაიდერების მხარეს. მომსახურების გაწევისთვის გადახდა ყოველთვიურად ხდება. მომხმარებელი იხდის მხოლოდ მოხმარებული რესურსებისთვის. ამ ტიპის მომსახურების ძირითადი უპირატესობებია:

- წინასწარ კონფიგურირებულ გარემოსთან თავისუფალი წვდომა;
- უკანასკნელი თაობის ინფრასტუქტურის გამოყენება;
- დაცული და იზოლირებული პლატფორმების გამოყენება;
- მესამე პირების მიერ მხარდაჭერილი ჩაშენებული რესურსების გამოყენების ხარჯზე რისკის შემცირება;
- პიკური დატვირთვების მართვის უნარი;
- შედარებით დაბალი ხარჯები;
- ფუნქციონალობის დამატების ან გაფართოების ნაკლები დრო, ღირებულება და სირთულეები.

მოთხოვნის მიხედვით გამოთვლები უფრო და უფრო მეტ პოპულარობას იძენს საწარმოებს შორის. გამოთვლითი რესურსები, რომლებიც ემსახურებიან მომხმარებელთა ვებ საიტებს, სულ უფრო და უფრო ნაკლები ხდება, მაშინ როცა მიმწოდებელთა ხელმისაწვდომი მომსახურების რესურსები მუდმივად იზრდება. მოთხოვნაზე ორიენტირებული მოდელი

განვითარდა სისტემების რესურსების მიმართ მერყევი მოთხოვნების ეფექტურად დაკმაყოფილების პრობლემის გადასაჭრელად. გამოთვლით რესურსების მიმართ მოთხოვნა შეიძლება მნიშვნელოვნად შეიცვალოს დროის საკმაოდ მცირე მონაკვეთის განმავლობაში და საკმარისი რესურსების მხარდაჭერა იმისათვის, რომ დაკმაყოფილდეს პიკური მოთხოვნა შეიძლება ძვირადღირებული იყოს. გადაწყვეტილების ტექნიკური გართულება შეიძლება ისევე არახელსაყრელი იყოს, როგორც სიტუაცია, როდესაც საწარმო ამცირებს ხარჯებს, და მხარს უჭერს მხლოდ მინიმალურ გამოთვლით რესურსებს. ისეთი ცნებები, როგორიცაა გამოთვლის კლასტერები, გამოთვლის გრიდი და ა.შ. შეიძლება მოგვეჩვენოს მსგავს ცნებებად მოთხოვნითი გმოთვლებისას, მაგრამ მათი უკეთ გაგება შეიძლება თუ მათზე ვიფიქრებთ, როგორც ისეთ სტანდარტულ ბლოკებზე, რომლები დიდი ხნის მანძილზე ვითარდებოდნენ იმისათვის რომ მიეღწიათ თანამედროვე ღრუბლური გმოთვლებისათვის, რომელსა ჩვენ დღეს ვიყენებთ.

ნახ.20. გამოთვლის გრიდი

Amazon

განვიხილოთ ერთ-ერთი მაგალითი - Amazon's Elastic Compute Cloud (Amazon EC2). *Amazon EC2* არის ვებ-სამსახური, რომელიც უზრუნველყოფს ღრუბელში საკმაო ზომის გამოთვლით სიმძლავრეებს. ის შემუშავებულია იმისთვის, რომ ვებ-გამოთვლები ხალმისაწვდომი იყოს შემმუშავებლებისათვის და რომ შეთავაზებული იყოს კლიენტებისათვის მრავალი უპირატესობა:

•ვებ-სამსახურის ინტერფეისი, მომხმარებელს საშუალებას აძლევს მიიღოს და შექმნას სივრცე მინიმალური ძალისხმევით;

• მომხმარებლებს გამოთვლითი რესურსებზე სრული კონტროლის უფლებას (იჯარით) და საშუალებას აძლევს. ნებას რთვს მათ მუშაობდნენ აპრობირებულ გამომთვლელ გარემოში;

•წუთებამდე ამცირებს დროს, რომელიც სჭირდება ახალი სერვისის მიღებას და ჩამოტვირთვას, რაც საშუალებას აძლევს კლიენტებს სწრაფად შეცვალონ კონფიგურაცია მათი გამოთვლითი მოთხოვნების შესაბამისად;

•ცვლის გამოთვლით ეკონომიკას, რომელიც საშუალებას აძლევს მომხმარებელს გადაიხადოს მხოლოდ გამოყენებულ რესურსების საფასური;

•უზრუნველყოფს დეველოპერებს იმ ინსტრუმენტებით, რომლებიც საჭიროა გაუმართაობის მიმართ ტოლერანტობის მქონე აპლიკაციების შესაქმნელად და საშუალებას აძლევს იზოლირებულნი იყვნენ საერთო გაუმართაობის სცენარებიდან.

Amazon EC2 წარმოადგენს გამომთვლელ გარემოს, რომელიც მომხმარებელს საშუალებას აძლევს, გამოიყენოს ვებ ინტერფეისი ოპერაციული სისტემის ერთი ან მეტი ეგზემპლიარის გასაშვებად საჭირო მომსახურების მიღებისა და მართვის მიზნით. მათ შეუძლიათ ისარგებლონ ქსელური შეღწევის მართვის უფლებით და იმდენი სისტემით, რამდენიც სჭირდებათ. *Amazon EC2*-ის გამოყენებისათვის კლიენტებმა ჯერ აუცილებლად უნდა შექმნან Amazon Machine Image (*AMI*). ეს გამოსხულება შეიცავს ვირტუალურ გამოთვლით გარემოში გამოყენებულ პროგრამებს, ბიბლიოთეკებს, მონაცემებს და კონფიგურაციის დაკავშირებულ პარამეტრებს. *Amazon EC2* გვთვაზობს წინასწარ კონფიგურებულ გამოსახულებებს, რომლებიც შექმნილია დაუყონებლივ გაშვებისათვის აუცილებელი თარგებით. როდესაც მომხმარებლებმა განსაზღვრავენ და ჩამოაყალიბებენ თავიანთ *AMI*-ის, ისინი იყენებენ *Amazon EC2*-ის ინსტრუნეტებს გამოსახულების Amazon S3-ში ჩასატვირთად. Amazon S3-არის საწყობი, რომელიც უზრუნველყოფს უსაფრთხო, საიმედო და სწრაფ წვდომას AMI კლიენტთან. სანამ მომხმარებელები შეძლებენ გამოიყენონ AMI, მათ უნდა გამოიყენონ Amazon EC2 ვებ ინტერფეისი უსაფრთხოებისა და ქსელის წვდომისთვის.

კონფიგურაციის დროს მომხმარებლები ირჩევენ რომელი ტიპის კატეგორია და რომელი ოპერაციული სისტემა სურთ გამოიყენონ. ხელმისაწვდომ კატეგორიათა ტიპები შეადგენენ ორ განსხვავებულ კატეგორიას: სტანდარტული პროცესორი ან მაღალი CPU პროცესორი.

აპლიკაციების უმეტესობის დაკმაყოფილება უმჯობესია სტანდარტული შემთხვევით, რომელშიც შედის პატარა, დიდი და ძალიან დიდი პლატფორმის ეგზემპლარები. High-CPU-ის შემთხვევაში გამოიყენება ცენტრალური პროცესორის პროპორციულად მეტი რესურსები, ვიდრე შემთხვევითი წვდომის მეხსიერება (RAM), რაც უფრო შეეფერება გადატვირთულ აპლიკაციებს. High-CPU პროცესორის შემთხვევაში შესარჩევად არის საშუალო და ძალიან დიდი პლატფორმები. იმის განსაზღვრის შემდეგ თუ რომელი ობიექტი გამოიყენონ კლიენტებს შეუძლიათ დაიწყონ, შეწყვიტონ და მონიტორინგი განახორციელონ თავიანთი AMI-ის იმდენ ეგზემპლარზე, რამდენიც საჭიროა გამოყენებითი პროგრამირების ვებ-სამსახურის (APIs) ინტერფეისებით სარგებლობისას ან მართვის სხვა ინსტრუმენტების დიდი მრავალფეროვნების დროს, რომლითაც მიმდინარეობს მომსახურება. მომხმარებლებს შეუძლიათ აირჩიონ სურთ თუა არა საბოლოო წერტილების სტატისტიკური IP მისამართების გამოყენებით გაუშვან აპლიკაცია მონაცემთა დამუშავების სხვადასხვა ცენტრებში, ამასთან ისინი იხდიან მხოლოდ რესურსების ფაქტობრივი მოხმარებისათვის. მომხმარებელს ასევე შეუძლია შეარჩიოს ხელმისაწვდომი AMI ბიბლიოთეკიდან. მაგალითად, თუ აუცილებელია ჩვეულებრივი Linux სერვერი, კლიენტებად შეიძლება იყოს ერთ-ერთი სტანდარტული Linux ნაკრებები AMIs.

არსებობს EC2 მომსახურების საკმაოდ ბევრი განსაკუთრებულობა, რომლებიც უზრუნველყოფენ საწარმოებისთვის მნიშვნელოვენ შეღავათებს. უპირველეს ყოვლისა, *Amazon EC2* უზრუნველყოფს ფინანსურ სარგებელს. კომპანია Amazon-ის მსხვილი მასშტაბის და დიდი კლიენტურის ბაზის გამო, ეს იაფი ალტერნატივაა სხვა შესაძლო გადაწყვეტილებეს შორის. დაწყების და მართვისათვის გაწეული ხარჯები გაყოფილია კლიენტთა დიდ რაოდენობას შორის, რაც ღირებულებას ნებისმიერი კლიენტისთვის სხვა ალტერნატივასთან შედარებით დაბალს ხდის. კლიენტები გამოთვლითი სიმძლავრეებში, რომელსაც ისინი ფაქტობრივად იყენებენ, იხდიან ძალიან დაბალ პროცენტს. უსაფრთხოება ასევე უზრუნველყოფილია *Amazon EC2* -ის ვებ-მომსახურების ინტერფეისების მეშვეობით. ეს ინტერფეისები საშუალებას აძლევენ მომხმარებელს მოახდინოს ბრანდმაუერის კონფიგურირების იმ პარამეტრების ფორმირება, რომლებიც აკონტროლებენ სამსახურის ეგზემპლარების ჯგუფებს შორის ქსელის ხელმისაწვდომობას. Amazon EC2 გთავაზობთ

ძალიან საიმედო გარემოს, სადაც ჩანაცვლების შემთხვევები სწრაფად შეიძლება იყოს უზრუნველყოფილი.

- დინამიური სკალარულობა.

Amazon EC2 მომხმარებელს საშუალებას აძლევს გაზარდოს ან შეამციროს პროდუქტიულობა რამდენიმე წუთში. მომხმარებელს შეუძლია გაუშვას ერთადერთი, ასობით ან ათასობით მომსახურების ეგზემპლიარი ერთდრულად. ყოველივე ეს იმართება ვებ-სერვისით API-ის გამოყენებით, აპლიკაციას შეუძლია თავისი თავის სკალირება ავტომატურად ზევით ან ქვევით, მისი მოთხოვნილების მიხედვით. ამ ტიპის დინამიური სკალარულობა ძალიან მიმზიდველია საწარმოს კლიენტებისათვის, რადგან იგი საშუალებას აძლევს მათ შეესაბამებოდნენ თავისი კლიენტების მოთხოვნებს საკუთარი ინფრასტუქტურის შევსების გარეშე.

- ეგზმპლიარებზე სრული კონტროლი.

მომხმარებლებს აქვთ სრული კონტროლი მათ ეგზემპლიარებზე. მათ აქვთ სრული წვდომა თითოეულ ეგზემპლიართან, შეუძლიათ მათზე ზემოქმედება ნებისმიერი მანქანიდან. ეგზემპლიარები შეიძლება იყოს გადატვირთული დაშორებული API ვებ-მომსახურების საშუალებით. მომხმარებლებს ასევე აქვთ თავიანთი ეგზემპლიარების კონსოლებთან წვდომა. როგორც კი მომხმარებელი მოახდენს თავისი ექაუნთის კონფიგურაციას და ჩატვირთავს თავის *AMI*-ს *Amazon S3* მომსახურებაში, მათთვის მხოლოდ აუცილებელია გაუშვას ეგზემპლიარი. *AMI*-ის გაშვება შეიძლება ეგზემპლიარების ნებისმიერი რიცხვის (ან ნებისმიერი ტიპისათვის) შესრულება RunInstances API- ის მეშვეობით, რომელსაც Amazon მხარს უჭერს.

- კონფიგურაციის მოქნილობა

კონფიგურაციის პარამეტრები შეიძლება მნიშვნელოვნად განსხვავდებოდეს მომხმარებლებს შორის. მათ აქვთ შესაძლებლობა არჩევანი გააკეთონ სხვადასხვა ტიპის ეგზემპლარებს, სისტემებს და პროგრამული უზრუნველყოფის პაკეტებს შორის. Amazon EC2 საშუალებას აძლევს მათ აირჩიონ მეხსიერების, ცენტრალური პროცესორის და შენახვის

სისტემის ის კონფიგურაცია, რომელიც ოპტიმალურია მათი ოპერაციული სისტემის შერჩევისა და გამოყენებისათვის. მაგალითად, მომხმარებლის მიერ არჩეული ოპერაციული სისტემა ასევე შეიძლება შეიცავდეს მრავალჯერად Linux builds, Microsoft Windows Server და კიდევ OpenSolaris, ყველა გაშვებულ ვირტუალურ სერვერს.

• *Amazon* -ის სხვა ვებ-სამსახურებთან ინტეგრაცია.

Amazon EC2 მუშაობს მრავალ სხვა Amazon ვებ სამსახურებთან ერთობლიობაში. მაგალითად, *Amazon Simple Storage Service (Amazon S3), Amazon SimpleDB, Amazon Simple Queue Service (Amazon SQS)* და *Amazon CloudFront* ყველა ინტეგრირებულები არიან იმისათვის, რომ უზრუნველყონ გამოთვლის, გამოთხოვნათა დამუშავების და შენახვის სრული გადაწყვეტა აპლიკაციების ფართო სპექტრს შორის.

Amazon S3 უზრუნველყოფს ვებ-სერვისების ინტერფეისს, რომელიც საშუალებას აძლევს მომხმარებლებს შეინახონ და აღადგინონ ნებისმიერი რაოდენობის მონაცემები ინტერნეტში ნებისმიერ დროს, ნებისმიერ ადგილას. ეს დეველოპერებს აძლევს პირდაპირი წვდომის საშუალებას იმავე, ყველაზე კარგად სკალარილებულ, საიმედო, სწრაფ და იაფ Amazon-ის მონაცემთა შენახვის ინფრასტუქტურაზე იმისათვის, რომ მართონ საკუთარი გლობალური ვებ-საიტების ქსელი. სერვისი S3 მიზნად ისახავს მაქსიმალურად გაზარდოს სკალაციის სარგებელიანობა და გადასცეს ეს უპირატესობა დეველოპერებს.

Amazon SimpleDB -ეს არის სხვა ვებ სერვისი, რომელიც იმისთვისაა შემუშავებული, რომ შეასრულოს გამოთხოვები *Amazon Simple Storage Service (Amazon S3* სტრუქტურული მონაცემების შესახებ) რეალური დროის რეჟიმში. ეს სერვისი მუშაობს Amazon EC2-სთან ერთობლიობაში იმისათვის, რომ მომხმარებლებს შენახვის, დამუშავების და მონაცემთა ნაკრების გამოთხოვების საშუალება მისცენ ღრუბლის გარემოს ფარგლებში. ეს სერვისები შემუშავებულია იმისათვის, რომ ვებ სკალაციური გამოთვლები ადვილი და მომხმარებლისათვის უფრო მისაღები გახადონ. ტრადიციულად, ფუნქციონირების ეს ტიპი უზრუნველყოფილი იყო კლასტერულ რელაციონური მონაცემთა ბაზის გამოყენებით, რომელიც მოითხოვს მნიშვნელოვნ ინვესტიციებს. ამ ტექნოლოგიების დანერგვამ უფრო მეტი სირთულე გამოიწვია და ხშირად მოითხოვდა ადმინისტირების მომსახურებას და მონაცემთა ბაზის შენარჩუნებას.

ტრადიციულ მიდგომასთან შედარებით, *Amazon SimpleDB* შესრულებაში მარტივია და უზრუნველყოფს ძირითად ფუნქციონირებას მონაცემების გარეშე (მაგალითად, რეალურ დროში ძებნა და სტრუქტურირებული მონცემების გამოთხოვება), რაც იწვევს ექსპლუატაციაში სიძნელეებს, რომლებიც წარმოიშობა ტრადიციული შესრულების დროს. *Amazon SimpleDB* არ საჭიროებს სქემებს, მონაცემების ინდექსირება ავტომატურად ხდება, უზრუნველყოფს მონაცემთა შენახვის და წვდომისათვის მარტივ API ინტერფეისს. ეს კლიენტებს თავიდან აშორებს იმ ამოცნების გადაწყვეტის აუცილებლობას, როგორიცაა მონაცემთა მოდერნიზება, ინდექსების მომსახურება და პროდუქტიულობის რეგულირება.

Amazon Simple Queue Service (Amazon SQS) - არის სერვისი, რომელიც იღებს შეტყობინებების რიგს შენახვისათვის. Amazon SQS-ის გამოყნებისას შემმუშავებლებს შეუძლიათ, შეტყობინების დაკარგვის გარეშე, უბრალოდ გადაადგილონ თავის აპლიაციების კომპონენტებში განაწილებული მონაცემები, რომლებიც ასრულებენ სხვადასხვა ამოცანას. ამასთან ამ დროს მიიღწევა მაღალი სკალარობა და საიმედოობა. Amazon SQS მუშაობს, როგორც Amazon-ის შეტყობინებების სკალარობის მასშტაბური გადაცემის ინფრასტუქტურის სერვისის დემონსტრირება. ნებისმიერი კომპიუტერს, რომელიც დაკავშირებულია ინტერნეტთან, შეუძლია დაამატოს ან წაიკითხოს შეტყობინება რაიმე პროგრამული უზრუნველყოფის ან სპეციალური კონფიგურაციის ფაიარვოლის დაყენების გარეშე. აპლიკაციების კომპონენტებს, რომლებიც Amazon SQS-ში გამოიყენება, შუძლიათ დამოუკიდებლად გაიშვნენ და არაა აუცილებელი განლაგდნენ იმავე ქსელში, იმავე გამოყენებულ ტექნოლოგიაში ან იმავე სამუშაო დროს.

Amazon CloudFront – კონტენტას (შინაარსის) მიწოდების ვებ-სერვისი. *Amazon CloudFront* ინტეგრირდება სხვა Amazon Web Services-თან. სერვისის მიზანს წარმოადგენს მისცეს საშუალება შემმუშავებლებს და საწარმოებს საბოლოო მომხმარებლისათვის მარტივი საშუალებით გაავრცელონ შინაარსი დაბალი შეფერხებით, მონაცემთა მიწოდების მაღალი სიჩქარით, ამასთან არა მოითხოვება რაიმე ვალდებულება. ობიექტის მოთხოვნები ავტომატურად გადამისამართდება უახლოეს მოსაზღვარე სერვერზე. ამგვარად, შინაარსი მიწოდებულია საკუთესო შესაძლო პროდუქტიულობით. მოსაზღვარე სერვერი იღებს მომხმარებლის კომპიუტერის მოთხოვნას და უკავშირედება სხვა კომპიუტერს, იძახებს ორიგინალურ სერვერს, სადაც აპლიკაცია მდებარეობს. როდესაც ორიგინალური სერვერი

ასრულებს მოთხოვნას, ის აგზავნის აპლიკაციის მონაცემს უკან მოსაზღვრე სერვერზე, რომელიც მონაცემებს უკან გადასცემს კლიენტის კომპიუტერს, რომელიც განახორციელებდა მოთხოვნას. სერვერი მოხმარებისათვის თავისუფალს არ წარმოადგენს.

პლატფორმა როგორც სერვისი (PaaS)

„ღრუბლოვანი“ გამოთვლების განვითარებამ მიგვიყვანა ისეთი პლატფორმების გამოჩენამდე, რომლებიც საშუალებას იძლევიან შევქმნათ და გავუშვათ ვებ-აპლიკაციები. *პლატფორმა როგორც სერვისი (Platform as a Service, PaaS)* - ეს არის ინტეგრირებული პლატფორმის წარდგენა ვებ-აპლიკაციის შემუშავებისათვის, ტესტირებისათვის, განლაგებისა და მხარდაჭერისათვის როგორც მომსახურებები, რომელიც ორგანიზებულია ღრუბლოვანი გამოთვლების კონცეფციის საფუძველზე.

მოდელი *PaaS* ქმნის ყველა პირობას ინტერნეტის ქსელიდან ხელმისაწვდომი ვებ-აპლიკაციების და მომსახურებების შექმნის და მიწოდების სრული სასიცოცხლო ციკლის შესანარჩუნებლად, რაც არ მოითხოვს შემქმნელებისათვის პროგრამული უზრუნველყოფის ჩატვირთვას ან ისტალაციას შემმუშავებლების, IT მენეჯერების ან საბოლოო მომხმარებლებისათვის. IaaS მოდელისაგან განსხვავებით, სადაც შემმუშავებლებს შეუძლიათ ოპერაციული სისტემების განსაზღვრული ეგზემპლარები დილეტანტური შექმნილი აპლიკაციების მეშვეობით, *PaaS* -ის შემმუშავებლები დაინტერესებულნი არიან მხოლოდ ვებ შემუშავებით და არ ზრუნავენ იმაზე, თუ რომელი ოპერაციული სისტემაა გამოყენებული. *PaaS* სერვისები მომხმარებელს საშუალებას აძლევს ყურადღება გაამახვილონ ინოვაციებზე, და არა რთულ ინფრასტრუქტურაზე. ორგანიზაციებს თავისი ბიუჯეტის მნიშვნელოვანი ნაწილი შეუძლიათ მიმართონ აპლიკაციების შექმნაზე, რომლებიც უზრუნველყოფენ რეალურ ფასეულობას, ინფრასტუქტურის მხრდაჭერის ნაცვლად. ამგვარად *PaaS* მოდელი ხსნის მასობრივი ინოვაციების ახალ ერას. ახლა მთელი მსოფლიოს შემმუშავებლებს შეუძლიათ მიიღონ შეღწევის უფლება შეუზღუდავ გმომთვლელ სიმძლავრეებთან. ნებისმიერ ადამიანს, რომელსაც აქვს ინტერნტთან წვდომა, შეუძლია შქმნას აპლიკაცია და ადვილად განათავსოს.

ლოკალური (On-*Premises*) აპლიკაციების შექმნის და გაშვების ტრადიციული მიდგომა ყოველთვის რთული, ძვირი და სარისკო იყო. თქვენი საკუთარი გადაწყვეტილების აგება არასოდეს არ წარმოადგენდა წარმატების გარანტიას. ყოველი აპლიკაცია შემუშავებული იყო

განსაზღვრული სამუშაო მოთხოვნების დასაკმაყოფილებლად. თითოეული გადაწყვეტილება მოითხოვდა აპრატორული საშუალებების, ოპერაციული სისტემების, მონაცემთა ბაზის, ელექტრონული ფოსტის, ვებ-სერვისების და ა.შ. განსაზღვრულ კონფიგურაციას. როცა შეიქმნა აპრატორული და პროგრამული უზრუნველყოფის გარემო შემმუშავებლების რაზმს უნდა აერჩია შემუშავებისათვის პლატფორმების კომპლექსი იმისათვის, რომ შეექმნა აპლიკაცია. რა თქმა უნდა ბიზნესი შემმუშვებლებისაგან მითხოვს აპლიკაციებში ცვლილებების განხორციელებას. აპლიკაციებში ცვლილებები, მის გავრცელებამდე მოითხოვს გამოსაცდელი ტესტის ჩატარების ახალ ციკლებს. მსხვილ კომპანიებს ხშირად სჭირდებათ სპეციალიზებული საშუალებები იმისათვის, რომ ისინი განათავსონ მონაცემთა დამუშავების ცენტრებში. სერვერების მუშობას და კონდიციონირების სისტემის მხარდაჭერას დიდი რაოდენობით ელექტროენერგია სჭირდება. საბოლოოდ, ყველაფერი ეს მოითხოვს გაუმართაობისადმი ტოლერანტობის მქონე ფართის გამოყენებას მონაცემთა დამუშვების ცენტრისთვის ისე, რომ გაუმართობის შემთხვევაში შესაძლებელი იყოს ინფორმაციის კოპირება.

PaaS გთავაზობთ აპლიკაციების დამუშვებისა და მიწოდებისათვის უფრო სწრაფ და ეკონომიკურად ეფექტურ მოდელის გამოყენებას. PaaS უზრუნველყოფს ინტერნეტის მეშვეობით აპლიკაციების გასაშვებად მთელ ინფრასტუქტურას. ანალოგიურ სერვისებს გვთავაზობენ კომპანიათა დიდი რაოდენობა, როგორებიცაა Microsoft, Amazon.com, Google. *PaaS*. PaaS დაფუძნებულია ლიცენზიების სააღრიცხვო ან სააბონენტო მოდელზე, ასე რომ მომხმარებლები იხდიან მხოლოდ იმაში, რასაც ისინი იყენებენ. *PaaS*-ის შემოთვაზება მოიცავს აპლიკაციების შექმნისათვის სამუშაო პროცესებს, აპლიკაციების დამუშავებას, ტესტირებას, განლაგებას და განთავსებას. ასევე აპლიკაციების სერვისს, ვირტუალურ ოფისებს, გუნდის თანამშრომლობას, მონაცემთა ბაზის ინტეგრაციას, უსაფრთხოებას, სკალარულობას, შენახვას, ოპერატიულობას, ქონების მართვას, ინსტრუმენტების პანელების აპარატურას და მრავალ სხვას.

PaaS-ის მთავარი თავისებურება მოიცავს სერვისებს აპლიკაციების შემუშავებისათვის, ტესტირებისათვის, განლაგებისათვის და მართვისათვის აპლიკაციების შემუშავების სასიცოცხლო ციკლის მხარდასაჭერად. შექმნის ხელსაწყოს ვებ ინტერფეისი, როგორც წესი, უზრუნველყოფს მხარდაჭერის გარკვეულ დონეს, რათა გაამარტივოს სამომხმარებლო ინტერფეისების შექმნა რომლებიც ისეთ ტექნოლოგიებზეა დაფუძნებული, როგორიცაა HTML,

JavaScript და სხვა ტექნოლოგიები. მრავალმომხმარებლიანი არქიტექტურის მხარდაჭერა თავიდან გვაცილებს შემუშავების დროს ერთდროულად ბევრი მომხმარებლის მიერ აპლიკაციების გამოყენების პრობლემებს. PaaS პროვაიდერები ხშირად მოიცავენ პარალლური დამუშავების, სკალურობის, გაუმათობისადმი ტოლერანტობის და უსაფრთხოების მომსახურებებს. კიდევ ერთი ფუნქცია, ეს არის ინტეგრაცია ვებ სერვისებსა და მონაცემთა ბაზებთან. განაწილებულ გამოთვლით გარემოში სტუქტურული შეტყობინებების პროტოკოლის (Simple Object Access Protocol, SOAP) მხარდაჭერა და სხვა ინტერფეისები PaaS-ის აპლიკაციებს საშუალებას აძლევს ვებ-სერვისების კომბინაციები (რომელსაც უწოდებენ mashup-ს) ისე ადვილად შექმნას, როგორც მონაცემთა ბაზების ხელმისაწვდომობის არსებობა და კერძო ქსელებში სერვისების ხელახალი გამოყენება. სპეციალიზირებული, წინასწარ განსაზღვრული ან განაწილებული ბრძანებების კოდექსის შექმნისა და გაცემის უნარი მნიშვნელოვნად ზრდის PaaS ვენდერების შეთავაზებებს. PaaS ინტეგრირებული შეთავაზება დეველოპერებს საშუალებას აძლევს უკეთ გაერკვნენ თავიანთი აპლიკაციების შიგა სამუშაოებში და მოწყობილობის პანელის მსგავს ინსტრუმენტების გამოყენებისას მომხმარებელთა ქცევაში იმისათვის, რომ განიხილონ შიგა პარამეტრები, რომლებიც ეფუძნება პარალელური შეერთებების რაოდენობის გაზომვაზე და ა.შ. ზოგიერთი PaaS შეთავაზება აფართოვებს ამ ინსტრუმენტარიებს, რაც საშუალებას იძლევა შევადგინოთ გამოყენებისათვის გადახდის ქვითრები.

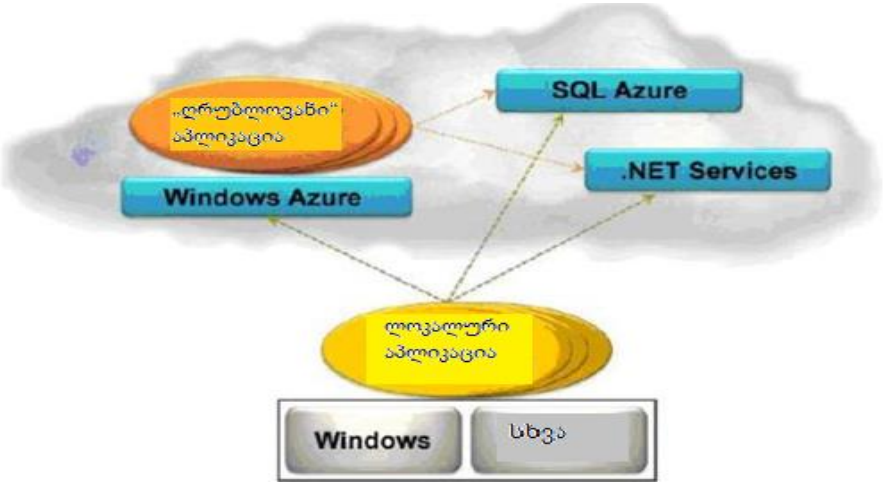
Microsoft Azure

კორპორაცია მაიკროსოფტ Windows Azure (თავდაპირველად ცნობილი იყო Azure Services Platform სახელით) - ეს არის „ღრუბლოვანი“ ტექნოლოგიის ჯგუფი, რომელთაგან თითოეული წარმოადგენს აპლიკაციის შემქმნელთათვის მომსახურების განსაზღვრულ ნაკრებს. ნახ.21-ზე ნაჩვენებია, რომ პლატფორმა Windows Azure შეიძლება იყოს როგორც ლოკალურ კომპიუტერებზე მომუშავე აპლიკაციები.

პლატფორმა Windows Azure შედგება შემდეგი კომპონენტებისაგან:

- Windows Azure. უზრუნველყოფს Windows-ზე დაფუძნებული გარემოს აპლიკაციების გაშვებისათვის და სერვერებზე მონაცემების შენახვისათვის Microsoft მონაცემთა ცენტრებში.
- SQL Azure. უზრუნველყოფს მონაცემთა მომსახურებას SQL Server-ის ბაზაზე „ღრუბელში“.

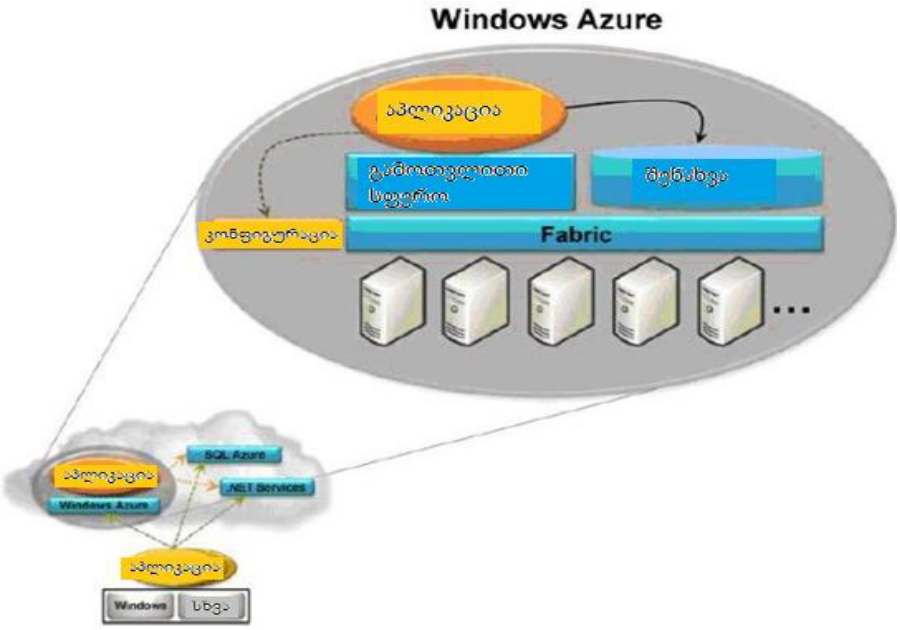
• .NET Services. უზრუნველყოფს ინფრასტუქტურის განაწილებას „ღრუბლოვანი“ აპლიკაციებისათვის და ლოკალური აპლიკაციებისათვის.



ნახ. 21. პლატფორმა Windows Azure მხარს უჭერს „ღრუბალში“ არსებულ აპლიკაციებს, მონაცემებს და ინფრასტუქტურას.

პლატფორმა Windows Azure -ის თითოეული კომპონენტი ასრულებს საკუთარ როლს.

Windows Azure -ის მაღალ დონეზე გაგება ძალიან მარტივია. ეს პლატფორმა Windows-ის აპლიკაციების შესრულებისთვის და მათი ინტერნეტში (ღრუბელში) შენახვისთვისაა. ნახ. 22 ნაჩვენებია მისი ძირითადი კომპონენეტები.



ნახ. 22. Windows Azure უზრუნველყოფს „ღრუბლოვან“ აპლიაკაციების მომსახურებას Windows-ის ბაზაზე გამოთვლისათვის და შენახვსათვის.

როგორც ნახატზეა ნაჩვენები, Windows Azure სრულდება დიდი რაოდენობის კომპიუტერებზე, რომლებიც განლაგებულია კორპორაცია მაიკროსოფტის მონაცემთა დამუშავების ცენტრებში და ხელმისაწვდომია ინტერნეტის საშუალებით. Fabric Windows Azure-ის დაკავშირების ზოგადი სტრუქტურა აკავშირებს ბევრ კომპიუტერულ სიმძლავრეს ერთ მთლიანობაში. Windows Azure სერვისები გამოთვლისა და შენახვისათვის ამ სტრუქტურაზეა აგებული.

Windows Azure-ის გამოთვლითი მომსახურება, რა თქმა უნდა, Windows-ის ბაზაზე მუშაობს. ამ მომსახურების საწყისი შეღწევდობის უზრუნველსაყოფად 2008 წლის შემოდგომაზე ფართე საზოგადოებისათვის გახსნილი იყო *CTP*-ვერსია. კორპორაცია მაიკროსოფტმა Windows Azure -ზე მხოლოდ იმ აპლიკაციების შესრულების ნებართვა გასცა, რომლებიც შემუშავებული იყო .NET Framework. პლატფორმაზე. მაგრამ ამჟამად, Windows Azure ასევე მხარს უჭერს უმართავ კოდს, რომელიც დეველოპერებს იმ აპლიკაციების გაშვების საშუალებას აძლევს, რომლებიც არ არიან დაფუძნებული .NET Framework.-ის ბაზაზე. ნებისმიერ შემთხვევაში ასეთი აპლიკაციები დაწერლია ჩვეულებრივ ენებზე Windows — C#, Visual Basic, C++ და სხვა Visual Studio 2008-ის მეშვეობით ან შემუშავების სხვა საშუალებებით. დეველოპერებს შუძლიათ ვებ-აპლიკაციების შექმნა ისეთი ტექნოლოგიების გამოყენებით, როგორიცაა ASP.NET და Windows Communication Foundation (WCF), ან ისეთი აპლიკაციების, რომლებიც იყენებენ დამოუკიდებელ ფონის პროცესებს, ან კიდევ აპლიკაციებს, რომლებიც აერთიანებენ ორივეს.

როგორც Windows Azure-ის აპლიკაციას, ასევე ლოკალურ აპლიკაციებს შეუძლიათ მიიღონ Windows Azure-ის საცავიდან შეღწევის უფლება ერთიდაიმავე საშუალებით: RESTful მიდგომის მეშვეობით. მაგრამ Microsoft SQL Server არ წარმოადგენს მონაცემთა საბაზო საცავს. ფაქტობრივად Windows Azure-ის საცავი არ მიეკუთვნება რელაციონურ სისტემებს, და მისი გამოთხოვების ენას არ წარმოადგენს SQL. რამდენადაც თავდაპირველად ის განკუთვნილი იყო Windows Azure-ის ბაზაზე აპლიკაციებისათვის, ამდენად ის უზრუნველყოფს შედარებით მარტივ და სკალირებულ შენახვის საშუალებებს. ამიტომ, ისინი დიდი ოროობითი ობიექტების (binary large object — blob) შენახვის საშუალებას იძლევიან, უზრუნველყოფენ აპლიკაციების კომპონენტებს შორის ზემოქმედებისათვის რიგების და ცხრილების მსგავს რაღაცის შექმნას გამოთხოვების მარტივი ენით.

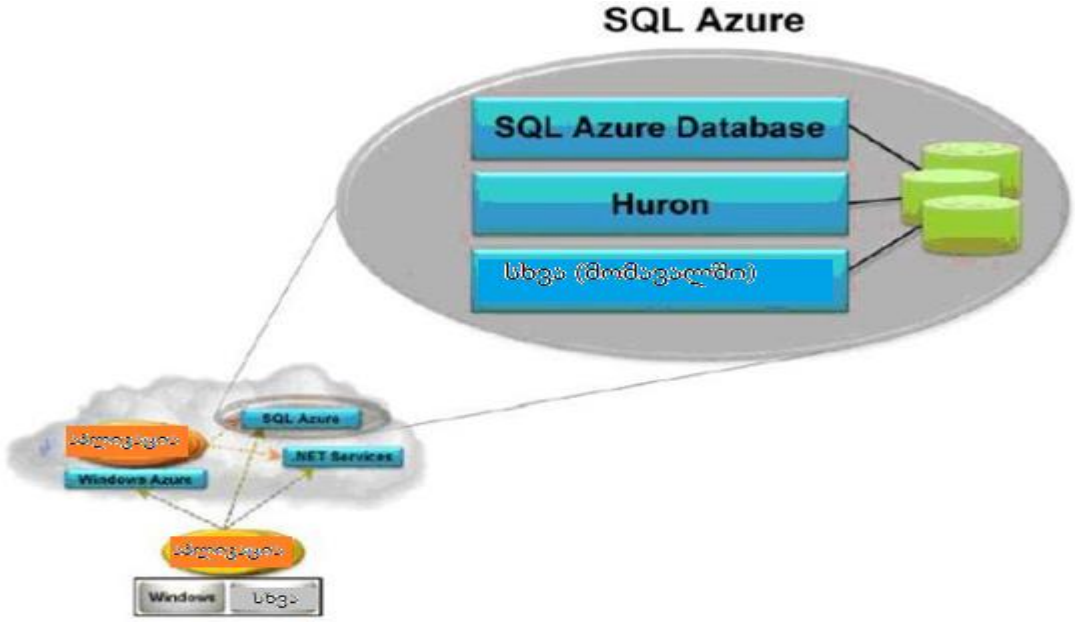
Windows Azure - ეს არის ზოგადი პლატფორმა, რომლის გამოყენებაც შესაძლებელია სხვადასხვა სცენარებში. მოვიყვანოთ რამდენიმე მაგალითი, ყველა მათგანი აღიწერება *CTP*-ვერსიის შესაძლებლობის გათვალისწინებით.

- ახალი ვებ-საიტის შესაქმნელად, ვთქვათ, ისეთის როგორიცაა Facebook, შეიძლება შემუშავდეს აპლიკაცია Windows Azure-ის პლატფორმაზე. იმის გამო, რომ მოცემული პლატფორმა მხარს უჭერს როგორც ვებ-სამსახურს ისე ფონურ პროცესებს, აპლიკაცია შეიძლება წარმოადგენდეს მომხმარებლის ინტერაქტიულ ინტერფეისს და ასინქრონულად ასრულებდეს სამუშაოს მომხმარებლისთვის. იმის მაგივრად რომ ინფრასტრუქტურაზე დახარჯოს ფული და დრო გამშვებმა ჯგუფმა შეიძლება მთლიანად ფოკუსირება მოახდინოს კოდის განვითარებაზე, რაც სარგებელს მოუტანს მომხმარებლებს და ინვესტორებს. კომპანიას ასევე შეუძლია გაუშვას პატარა ვებსაიტი, რომელიც მცირე ხარჯს მოითხოვს, თუ ამ აპლიკაციას ძალიან ცოტა მომხმარებელი ჰყავს. თუ აპლიკაცია იძენს პოპულარობას და მომხმარებელთა რიცხვი იზრდება, Windows Azure საჭიროების შემთხვევაში ახდენს მის სკალარულობას.
- პროგრამული უზრუნველყოფის დამოუკიდებელმა მიმწოდებლებმა, რომლებიც ქმნიან პროგრამების ლოკალური Windows აპლიკაციის მქონე მომსახურებების ვერსიის სახით, შეუძლიათ ის შეიმუშაონ Windows Azure-ის ბაზაზე. იმის წყალობით, რომ Windows Azure ძირითადად უზრუნველყოფს Windows-ის სტანდარტულ გარემოს ბიზნეს-ლოგიკა ამ "Cloud" პლატფორმაზე გადანაცვლებამ არ უნდა გამოიწვიოს რაიმე განსაკუთრებული პრობლემები. კიდევ ერთხელ გავუსვათ ხაზი: არსებულ პლატფორმის ბაზაზე შემუშავება პროგრამული უზრუნველყოფის დამოუკიდებელ მიმწდებლებს საშუალებას აძლევს ყურადღების ფოკუსირება მოახდინონ ბიზნეს-ლოგიკაზე, ე.ი. იმაზე, რაც ფულის კეთების საშუალებას იძლევა და არ დახარჯონ დრო ინფრასტუქტურაზე.
- კომპანია, რომელიც ქმნის თავის მომხმარებლისთვის აპლიკაციებს, მისი შემუშავებისათვის შეუძლია აირჩიოს Windows Azure პლატფორმა. იმის გამო, რომ Windows Azure .NET-ს მხარს უჭერს, სირთულეს არ წარმოადგენს მოიძებნოს შემმუშავებელი შესაბამისი უნარებით ამასთან გონივრული ანაზღაურებით. Microsoft მონაცემთა დამუშავები ცენტრებში აპლიკაციების შესრულება ათავისუფლებს საწარმოებს პასუხიმგებლობისაგან და საკუთარი სერვერების მხარდაჭერაზე გაწეული ხარჯისაგან, რომელიც კაპიტალურ ხარჯებს აქცევს

საექსპლუატაციო ხარჯებად. კერძოდ, თუ აპლიკაციას აქვს პიკური დატვირთვის პერიოდი, Microsoft კორპორაციის მიერ შეთავაზებული დიდი სერვერების ბაზის ფუნქციის მხარდაჭერა შეიძლება ეკონომიკურად მომგებიანი აღმოჩნდეს.

„ღრუბელში“ აპლიკაციების შესრულება - „ღრუბლოვანი“ გამოთვლების ერთ-ერთი მნიშვნელოვანი ასპექტია. Windows Azure-ის დახმარებით კორპორაცია Microsoft უზრუნველყოფს როგორც პლატფორმას გამოთვლისათვის, ისე მონაცემთა შენახვის საშუალებებს. იმისდა მიხედვით, თუ როგორ იზრდება ინტერესი „ღრუბლოვანი“ გამოთვლების მიმართ, მოსალოდნელია კიდევ უფრო მეტი რაოდენობის Windows აპლიკაციების შექმნა ამ ახალი სფეროსთვის.

ინტერნეტის მეშვეობით ხელმისაწვდომი სერვერების გამოყენების ერთ-ერთი მიმზიდველი საშუალებაა მონაცემთა დამუშავება. SQL Azure-ის მიზანს ამ პრობლემის გადაწყვეტა წარმოადგენს, გვათავაზობს რა შენახვისათვის ვებ-მომსახურების ნაკრებს სხვადასხვა ინფორაციისათვის და მათზე სამუშაოდ. მაშინ როდესაც Microsoft-ის წარმომადგენლებმა განაცხადეს, რომ SQL Azure თანდათნ მოიცავს მთელ რიგ შესაძლებლობებს, რომლებიც მონაცემებზე იქნებოდა ორიენტირებული, მათ შორის: ანგარიშების შექმნა, მონაცემთა ანალიზი და ბევრი სხვა, SQL Azure-ის პირველი კომპონენტები გახდება SQL Azure Database-ის მონაცემთა ბაზა და მონაცემთა სინქრონიზაციის საშუალება Huron . ეს თავლსაჩინოდაა გამოსახული ნახ. 23-ზე.



ნახ. 23. SQL Azure უზრუნველყოფს ინტერნეტში საშუალებებს, როლებიც ორიენტირებულები არიან მონაცემებთან მუშაობაზე.

მონაცეთა ბაზა SQL Azure Database (ადრე ცნობილი იყო SQL Data Services სახელწოდებით) უზრუნველყოფს ინტერნეტში მონაცემთა ბაზების მართვის სისტემას (მბმს), ეს ტექნოლოგია ლოკალურ და ვებ-აპლიკაციებს რელაციონალურ და სხვა ტიპის მონაცემების Microsoft მონაცემთა დამუშავების ცენტრებში Microsoft სერვერებზე შენახვის საშუალებას აძლევს. ისევე როგორც სხვა ვებ-ტექნოლოგიებთან მუშაობისას, კომპანია იხდის მხოლოდ იმაში რასაც გამოიყენებს, ზრდის და ამცირებს გამოყენებულ მოცულობას (და ხარჯებს) ცვლილებებში წარმოშობილ აუცილებლობიდან გამომდინარე. „ღრუბელში“ მონაცემთა ბაზის გამოყენება ასევე ცვლის კაპიტალური დანახარჯის ხასიათს: მყარ დისკებში და პროგრამულ უზრუნველყოფაში ხარჯის მაგივრად, მონაცემთა ბაზების მართვის სისტემისათვის ექსლუტაციური ხარჯების გაწევა ხდება.

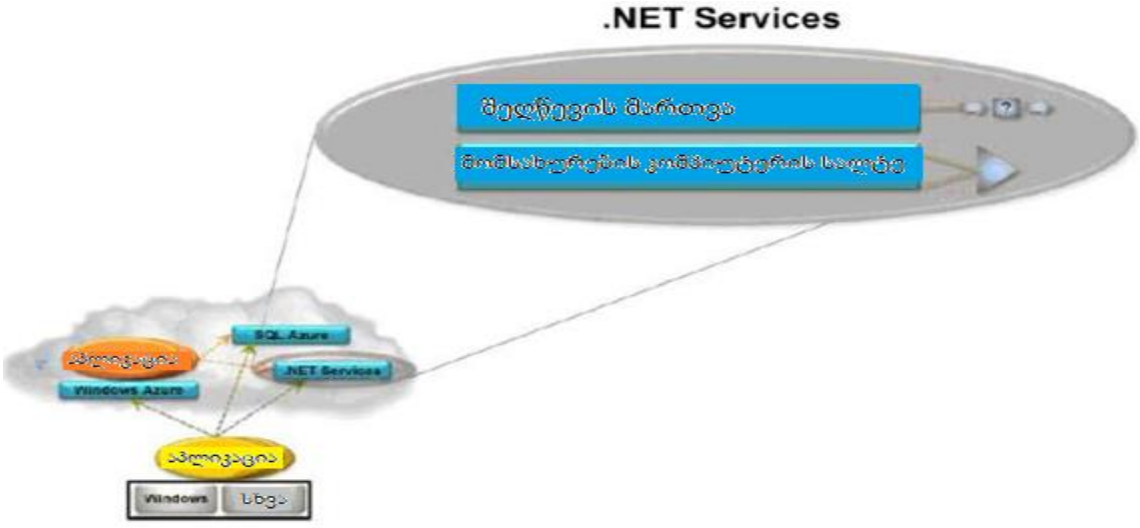
Windows Azure-ის საცავის მომსახურებისაგან განსხვავებით SQL Azure მონაცემთა ბაზა შემუშავებულია Microsoft SQL Server-ის საფუძველზე. მიუხედავად ამისა თავდაპირველ *CTP* 2008 წლის ვერსიაში მონაცემთა ბაზა SQL Azure Database არ წარმოადგენდა მონაცემების მიმართ ტრადიციულ რელაციურ მიდგომას. შემკვეთთა გამოხმაურებების გათვალისწინებით, კორპორაცია Microsoft-მა გადაწყვიტა შეეტანა შესაბამისი ცვლილებები. სამომავლოდ მონაცემთა ბაზა SQL Azure Database მხარს დაუჭერს რელაციურ მონაცემებს, უზრუნველყოფს რა SQL Server გარემოს „ღრუბელში“ ინდექსებით, წარმოდგენებით, შესანახი პროცედურებით, ტრიგენებით და სხვა მრავალით. ამ მონაცემებზე წვდომა შეძლება მივიღოთ ADO.NET -ის და სხვა Windows მონაცემებთან წვდომის ინტერფეისების საშუალებით. ფაქტობრივად აპლიკციები რომელბიც ახლა ლოკალურად იღებენ SQL Server-თან წვდომას, SQL Azure Database-ში მონაცემებთან იმუშავებენ თითქმის იმნაირადვე. „ღრუბელში“ ამ ინფორმაცისთან მუშაობისათვის შემკვეთებს შეუძლიათ გამოიყენონ ლოკალური პროგრამული უზრუნველყოფა, ისეთის როგორიცაა SQL Server ანგარიშგების მომსახურება.

მიუხედავად იმისა, რომ პროგრამებს შეუძლიათ გამოიყენონ SQL Azure Database მონაცემთა ბაზა, ისევე, როგორც ადგილობრივი მონაცემთა ბაზების მართვის სისტემა, მართვის მოთხოვნები არსებითად შემცირდა. იმის მაგივრად თვალყური ადევნოს ტექნიკას, მაგალითად, უზრუნველყოს დისკის გამოყენების მონიტორინგი და ფაილების ჟურნალის მომსახურება, SQL Azure Database-ის შემკვეთს შეუძლია მთელი ყურადღების კონცენტრაციას მოახდინოს იმაზე, რაც მართლაც მნიშვნელოვანია, მონაცემებზე. კორპორაცია Microsoft

ყურადღებას მიაქცევს ექსპლუატაციის საკითხებს. გარდა ამისა, ისევე როგორც პლატფორმა Windows Azure -ის ხვა კოპონენტები შემთხვევაში, SQL Azure Database-ის გამოყენება არ წარმოადგენს სირთულეს. საჭიროა მხოლოდ ვებ-პორტალზე შესვლა და საჭირო ინფორმაციის წარმოდგენა.

პროგრამების შესრულება და ინტერნეტში მონაცემთა შენახვა, გამოთვლითი ქსელური გარემოს მნიშვნელოვან ასპექტს მიეკუთვნება. მაგრამ ისინი მთლიანად არ ამოწურავენ მის შესაძლებლობას. სხვა შესაძლებლობას წარმოადგენს ინფრასტუქტურის უზრუნველყოფაში „ღრუბლის“ ბაზაზე მომსახურება, რომელიც შეიძლება გამოყენებული იყოს ლოკალური აპლიკაციებით ანა ვებ-აპლიკაციებით. ამ უფსკრულის ამოსავსებადაა მოწოდებული .NET Services. მომსახურებები.

.NET Services. მომსახურებები, რომლებიც თავიდან ცნობილი იყო როგორც BizTalk Services, გავთავაზობენ განაწილებადი აპლიკაციის შექმნისას ინფრასტრუქტურის ზოგადი პრობლემების გადაწყვეტისათვის ფუნქციებს. ნახ. 24-ზე ნაჩვენებია მათი ძირითადი კომპონენტები.



ნახ. 24 .NET Services. მომსახურებები „ღრუბელში“ უზრუნველყოფენ ინფრასტრუქტურას, რომლის გამოყენებაც შეიძლება ვებ-აპლიკაციებისა და ლოკალური აპლიკაციებისათვის.

.NET Services. მომსახურებები შდგება შემდეგი კომპონენტებისაგან.

- შეღწევის მართვა. იდენტობისადმი სულ უფრო ფართედ გავრცელებული მიდგომა იმაში მდგომარეობს, რომ თითოეულმა მომხმარებელმა უნდა მიაწოდოს აპლიკაციას მარკერი, რომელიც შეიცავს გარკვეული დადასტურებების კომპლექტს. ამ დადასტურებების

საფუძველზე აპლიკაცია გადაწყვეტს რისი გაკეთების უფლება აქვს მომხმარებელს. ამ პროცედურის ეფექტიანი განხორციელება კომპანიის მასშტაბში მოითხოვს სახელმწიფოს მოწმობას, რომელიც საშუალებას გაძლევთ მიიღოთ დამოწმება ერთ-ერთ ტერიტორიაზე გაკეთებული დადასტურებისა სხვა ადგილას. ასევე შეიძლება საჭირო გახდეს იმ ცვლილების დამტკიცება, რომელიც დადასტურებების ერთი ადგილიდან სხვა ადგილას გადატანისას ხდება. წვდომის მართვის სამსახური უზრუნველყოფს "ღრუბლის" საფუძველზე ორივე ფუნქციის განხორციელებას.

•სერვისების კომპიუტერის სალტე. ინტერნეტში აპლიკაციების მომსახურების გაწევა ბევრად უფრო რთულია, ვიდრე შეიძლება მოგვეჩვენოს. სერვისების კომპიუტერის სალტეს ამოცანაა ამ პროცედურის გამარტივება, რომელიც რომელიც საშუალებას აძლევს აპლიკაციებს, უზრუნველყოს ვებ-სერვისების საბოლოო წერტილები, რომლებზეც წვდომა შეიძლება მიღებული იყოს სხვა აპლიკაციებით - ლოკალურით ან „ღრუბელში“ მომუშავეთი. თითოეული გათვალისწინებული საბოლოო წერტილს ენიჭება URI, რომელიც კლიენტებს შეუძლიათ გამოიყენონ მომსახურების მოსაძებნად და მასზე შეღწევისთვის. სერვისის კომპიუტერის სალტე წარმოდგენილი აპლიკაციებისათვის ასევე წყვეტს ქსელის მისამართების კონვერტაციის პრობლემას და ქსელებს შორის ეკრანის გავლას ახალი პორტების გახსნის გარეშე.

მოვიყვანოთ .NET Services. მომსახურების გამოყენების რამდენიმე მაგალითი.

• პროგრამული უზრუნველყოფის დამოუკიდებელი მიმწოდებელს, რომლიც აწვდის სხვადასხვა ორგანიზაციიდან შემკვეთებს მათთვის საჭირო აპლიკაციებს შეუძლია გამოიყენოს შეღწევის მართვის მომსახურება აპლიკაციბის შემუშავების და ექსპლუატაციისათვის. მაგალითად, .NET Services.-ის ამ კომპონენტს შეუძლია გარდაქმნას დამოუკიდებელი პროგრამული უზრუნველყოფის მიმწოდებლის გამოყენებისათვის შესაფერისი სხვადასხვა დადასტურება, რომლებიც გამოიყენება სხვადასხვა ორგანიზაციებში სხვადასხვა საიდენტიფიკაციო ტექნოლოგიებით შეთანხმებულ ნაკრებში. ასეთი მოთხოვნა შესაძლებელს ხდის ასევე განიტვირთოს სახელმწიფოს მოწმობის მექანიზმი შეღწევის მართვის ხარჯზე, რომელიც ათავისუფლებს პროგრამული უზრუნველყოფის დამოუკიდებელ მიმწოდებლებს სახელმწიფოს მოწმობის საკუთარი პროგრამის შესრულების აუცილებლობისაგან.

• დავუშვათ, საწარმოს სურს შესაძლებელი გახადოს თავის სავაჭრო პარტნიორებისათვის ერთ-ერთ აპლიკაციაზე შეღწევა. მას შეუძლია აპლიკაციის ფუნქციები გაანაწილოს SOAP ან RESTful ვებ-მომსახურებების საშუალებით და დაარეგისტრიროს მათი საბოლოო წერტილები სერვისების კომპიუტერის სალტეს მეშვეობით. შემდეგ სავაჭრო პარტნიორები გამოიყენებენ კომპიუტერის სალტეს საბოლოო წერტილის და მომსახურებებზე წვდომის მოსაძებნად. ეს აპლიკაციების წარდგენასთან დაკავშირებული რისკის შემცირების საშუალებას იძლევა, რამდენადაც არ მოითხოვს ახალ იმპორტების გახსნას კომპანიის ქსელთაშორის ეკრანში. ორგანიზაციას ასევე შეუძლია ისარგებლოს წვდომის მართვის სამსახურით, რომელიც განკუთვნილია სერვისის სალტესთან სამუშაოდ, მისი პარტნიორების მიერ განაცხადზე გაგზავნილი აუთენტიფიკაციის გადამოწმების ინფორმაციის მიღების გამარტივებისათვის.

ისევე, როგორც Windows Azure-ის შემთხვევაში, წარმოდგენილია პორტალი, რომელიც ხელმისაწვდომია ბრაუზერის მეშვეობით, რათა მომხმარებელს საშუალება მიეცეს გამოიყენოს .NET Services. მომსახურება Windows Live ID.-ის გამოყენებით. კორპორაცია Microsoft -ის მიზანი, რომელიც მიიღწევა.NET Services.-ის მეშვეობითა სავსებით ცხადია: განაწილებული აპლიკაციებისათვის სასარგებლო "ღრუბელის" ინფრასტრუქტურის უზრუნველყოფა.

პროგრამული უზრუნველყოფა როგორც სერვისი (SaaS)

პროგრამული უზრუნველყოფა როგორც სერვისი (*Software as a service, SaaS*), ანუ პროგრამული უზრუნველყოფა მოთხოვნით (Software on Demand, *SoD*) - ეს არის პროგრამული უზრუნველყოფის გაყიდვის მოდელი, რომლის დროსაც მიმწოდებელი შეიმუშავებს ვებ-აპლიკაციას და დამოუკიდებლად მართავს მას, უზრუნველყოფს რა მომხმარებლებისთვის ინტერნეტის საშუალებით პროგრამული უზრუნველყოფაზე ხელმისაწვდომობას. მომხარებლისათვის SaaS მოდელის ძირითად უპირატესობას დაყენებასთან, განახლებასთან და მუშა მდგომარეობის მხარდაჭერასთან და მასზე მომუშავე პროგრამულ უზრუნველყოფასთან დაკავშირებული დანახარჯის არ ქონა წარმოადგენს. პროგრამული უზრუნველყოფა, როგორც სერვისი წარმოადგენს *პროგრამული უზრუნველყოფის განაწილების* მოდელს, რომელშიც აპლიკაციები განლაგებულია SaaS ვენდორთან ან მომსახურებლის მიმწოდებელთან და ხელმისაწვდომია კლიენტებისათვის ქსელში, როგორც წესი, ინტერნეტში. აპლიკაციების მიმწოდებელი მოდელი SaaS ხდება უფრო და უფრო

გავრცელებული ტექნოლოგია, რომელიც მხარს უჭერს ვებ-მომსახურებებს და სერვის-ორიენტირებულ არქიტექტურას (SOA). SaaS ასვე ხშირად ასოცირდება ლიცენზირების მოდელთან, როდესაც გადახდა მიმდინარეობს მიღებული მომსახურების მიხედვით. ამავდროულად, ფართეზოლიანი ქსელის მომსახურებები მთელს მსოფლიოში მომხმარებლებისათვის უფრო და უფრო ხელმისაწვდომები გახდნენ.

ინტერნეტ მომსახურების მიმწოდებლების (ISPs) მიერ მიღწეული უზარმაზარი წარმატება აშკარაა იმისთვის, რომ გაიზარდოს გატარების სიჩქარე და შეინახოს უფრო ძლიერი მიკროპროცესორების გამოყენების შესაძლებლობა, იაფი შენახვის მოწყობილობებთან ერთად. ეს უზრუნველყოფს უზარმაზარ პლატფორმის შექმნას იმისათვის, რომ დაპროექტდეს, დაინერგოს და გამოყენებული იყოს პროგრამული უზრუნველყოფა ბიზნესის ყველა სფეროში და კერძო გამოთვლების კუთხით. SaaS-ის აპლიკაციებს ასევე უნდა შეეძლოს სხვადასხვა გარემოსა და პლატფორმების ფართო სპექტრის სხვა მონაცემებთან და სხვა აპლიკაციებთან ურთიერთქმედება. კომპანია IDC აღწერს SaaS მოდელის ორ ოდნავ განსხვავებულ მიწოდებას.

SaaS უფრო ხშირად განკუთვნილია, კორპორაციული კლიენტებისათვის პროგრამული უზრუნველყოფის დაბალ ფასად მიწოდებისათვის, რაც კომპანიაში საშუალებას იძლევა თავიდან იქნეს აცილებული დამონტაჟება, მართვა, მხარდაჭერა, ლიცენზირებასთან დაკავშირებული მაღალი დანახარჯი. კლიენტების უმეტესობას არ აინტერესებს იცოდეს როგორ და რატომაა პროგრამული უზრუნველყოფა რეალიზებული, გაშლილი და ა.შ., მაგრამ ამავე დროს ყველას აქვს პროგრამული უზრუნველყოფის გამოყენების მოთხოვნა თავიანთ საქმიანობაში. პროგრამული უზრუნველყოფის მრავალი ტიპი კარგად აკმაყოფილებს SaaS მოდელებს (მაგალითად, საბუღალტრო აღრიცხვა, კლიენტებთან მუშაობა, ელექტრული ფოსტა, შრომითი რესურსების აღრიცხვა, IT უსაფრთხოება, IT მართვა, ვიდეოკონფერენცკავშირი, *ვებ-ანალიტიკა*, ვებ-კონტენტის მართვა). განსხვავება SaaS-სა და ინტერნეტით აპლიკაციის მიწოდების ადრინდელ საშუალებას შორის იმაში მდგომარეობს, რომ SaaS გადაწყვეტიებები სპეციალურად იყო შემუშავებული ვებ ბრაუზერებთან სამუშაოდ. SaaS-ის საფუძველზე აპლიკაციის არქიტექტურა სპეციალურად განკუთვნილია დიდი რაოდენობის მომხმარებლის მიმართვების მხარდასაჭერად. ამაში მდგომარეობს დიდი განსხვავება ტრადიციულ კლიენტი-სერვერული აპლიკაციის გადაწყვეტებს შორის, რომელიც მდებარეობენ მომსახურების მიმწოდებლებთან. მეორეს მხრივ, SaaS-ის მომსახურების

მიმწოდებლები ზრდიან სკალარულობის ეკონომიას, მათი აპლიკაციების განლაგების მართვისა, მხარდაჭერის და მომსახურების დროს.

SaaS-ის აპლიკაციების შემუშავებისას შეიძლება გამოყენებულ იქნას პროგრამული უზრუნველყოფის მრავალი ტიპის კომპონენტი და ფრეიმვორკი. ამ თანამედროვე კომპონენტებში და აპლიკაციების შემუშავების გარემოებებში ახალი ტექნლოგიების გამოყენებით შეიძლება მნიშვნელოვნად შემცირდეს ტრადიციული პროდუქტის SaaS გადაწყვეტაში ტრანსფორმაციის დრო და ღირებულება. Microsoft-ის თანახმად, SaaS არქიტექტურის კლასიფიცირება შეიძლება საკვანძო ნიშნების მიხედვით ოთხიდან ერთ-ერთ დონედ: კონფიგურაციების სამარტივე, მრავალმომხმარებლიან შეღწევადობის და სკალარულობის დროს ეფექტურობა. თითოეული დონე წინა დონისაგან განსხვავდება ერთ-ერთი ამ ნიშნის დამატებით. განვიხილოთ Microsoft-ის მიერ აღწერილი დონეები:

- **არქიტექტურული დონე 1** - სპეციალური / დაკონფიგურირებადი.

პირველი დონე ფაქტობრივად ყველაზე დაბალია, თითოეულ კლიენტს აქვს განთავსებული აპლიკაციის უნიკალური, დაკონფიგურირებული ვერსია. აპლიკაცია უშვებს საკუთარ ეგზემპლიარებს სერვერებზე. SaaS-ის ამ დონეზე ტრადიციული არაქსელური ან კლიენტ-სერვერული აპლიკაციების მიგრაცია, როგორც წესი, მოითხოვს მცირე ძალისხმევას შემუშავების დროს და ამცირებს საექსპლუატაციო ხარჯებს, აპარატორული უზრუნველყოფის და ადმინისტრირების გაერთიანების ხარჯზე.

- **არქიტექტურული დონე 2** - კონფიგურაცია.

SaaS-ის მეორე დონე უზრუნველყოფს პროგრამის დიდ მოქნილობას კონფიგურაციის მეტამონაცემების წყალობით. ამ ეტაპზე კლიენტებს შეუძლიათ გამოიყენონ იგივე აპლიკაციის მრავალი ცალკეული ეგზემპლიარი. ეს საშუალებას აძლევს ვენდორებს, დეტალური კონფიგურაციის გამოყენებისას, დააკმაყოფილონ თითოეული მომხმარებლის სხვადასხვა მოთხოვნა. ასევე, გამარტივებულია მომსახურება, შესაძლებელი ხდება საერთო კოდური ბაზის განახლება.

- **არქიტექტურული დონე 3** - მულტიარენდატორის ეფექტურობა.

მესამე დონე მეორისაგან განსხვავდება იმით, რომ მასში არის მრავალმომხმარებლიანი შეღწევადობის მხარდაჭერა. პროგრამის ერთადერთ ეგზემპლიარს შეუძლია მოემსახუროს ყველა მომხმარებელს. ეს მიდგომა საშუალებას გაძლევთ საბოლოოდ მომხმარებლისთვის

შეუმჩნევლად უფრო ეფექტურად გამოიყენოთ სერვერის რესურსები მაგრამ, საბოლოო ჯამში, ეს დონე არ იძლევა სისტემის მასშტაბირების საშუალებას.

• **არქიტექტურული დონე 4**– სკალარულობა.

SaaS-ის მეოთხე დონეში სკალარულობა დამატებულია მრავალდონიანი არქიტექტურის გამოყენების წყალობით. ამ არქიტექტურას შეუძლია აპლიკაციების იდენტური ეგზემპლიარების ფერმის დატვირთვის განაწილების მხარდაჭერა, რომლებიც გაშვებულია სერვერების ცვლად რაოდენობაზე, რომელთა რაოდენობა ასს ან ათასსაც კი აღწევს. სისტემის სიმძლავრე მოთხოვნების შესაბამისად დინამიურად შეიძლება გაიზარდოს ან შემცირდეს. ეს ხდება სერვერების დამატება ან მოხსნის გზით გამოყენებითი პროგრამული უზრუნველყოფის არქიტექტურის შემდგომი ცვლილების გარეშე.

სერვის-ორიენტირებული არქიტექტურაში აპლიკაციების განლაგება უფრო რთული საკითხია, ვიდრე ტრადიციულ მოდელებში პროგრამული უზრუნველყოფის განლაგება. შედეგად, SaaS აპლიკაციის გამოყენების ღირებულება ეფუძნება მომხმარებელთა იმ რაოდენობას, რომლებიც ახორციელებენ სერვერებთან წვდომას. საკმაოდ ხშირად წარმოიქმნება დამატებითი ხარჯები, რომლებიც დაკავშირებულია სერვისული მომსახურებების გამოყენებასთან, გაშვების დამატებით ზოლთან, და დამატებით დისკურ სივრცესთან. SaaS მომსახურების პროვაიდერების შემოსავლები ჩვეულებრივ თავდაპირველად დაბალია, ვიდრე ტრადიციული ხარჯები პროგრამული უზრუნველყოფის ლიცენზიებზე. თუმცა პროგრამული უზრუნველყოფის ლიცენზიაზე შედარებით დაბალი ხარჯის კომპრომისი არის შემოსავლის ყოველთვიური დაბრუნება, რომელიც განიხილება კომპანიის ფინანსური დირექტორის მიერ, როგორც ბიზნესის *არსებობისთვის* უფრო პროგნოზირებადი *კრიტერიუმი.* SaaS პროგრამული უზრუნველყოფის ძირითად თავისებურებებს მიეკუთვნება:

• ქსელის მართვა და ქსელური ხელმისაწვდომობა კომერციული პროგრამული უზრუნველყოფის ცენტრალიზებულ მონაცემთა დამუშავების ცენტრებში, და არა კლიენტების საიტებზე, რაც საშუალებას აძლევს მომხმარებელს, მიიღონ შეღწევის უფლება აპლიკაციებზე დისტანციურად, ინტერნეტის საშუალებით.

• აპლიკაციების მიწოდება, "ერთი-ბევრი" მოდელით, ტრადიციული მოდელისაგან "ერთი-ერთი" განსხვავებით.

- ცენტრალიზებული მოდერნიზაცია და განახლება, რაც თავიდან გვაცილებს მომხმარებელთა აპლიკაციების ჩამოტვირთვის და დაყენების აუცილებლობას. SaaS ხშირად გამოიყენება კომუნიკაციის დიდ ქსელებისა და პროგრამულ უზრუნველყოფაში ერთობლივი მუშაობისათვის, ზოგჯერ კი როგორც PaaS არქიტექტურის პროგრამული გაფართოება.

კომპანიებში პროგრამების შემუშავების ციკლმა შეიძლება საკმაოდ დიდი დრო დაიკავოს, დასჭიდეს ბევრი რესურსი და არადამაკმაყოფილებელ შედეგამდე მიგვიყვანოს. მიუხედავად იმისა, რომ კონტროლის დათმობის გადაწყვეტილების მიღება რთულია, ამან შეიძლება გამოიწვიოს გაუმჯობესებული ეფექტურობა, შემცირებული რისკები და შემცირებული ხარჯები. კომპანიების რაოდენობა, რომელსაც სურს გამოიყენოს SaaS მოდელი საწარმოს აპლიკაციებისათვის, როგორებიცაა კლიენტებთან მუშაობა, ფინანსური ხარჯები, პერსონალის მართვა, მუდმივად იზრდება. SaaS მოდელი საწარმოებს გარანტიას აძლევს, რომ სისტემის ყველა მომხმარებელი იყენებს აპლიკაციის სწორ ვერსიას და, შესაბამისად, რეგისტრირებული და გადაცემული მონაცემების ფორმატი სწორი, თავსებადი და ზუსტია. SaaS-ის პროვაიდერზე აპლიკაციების პასუხისმგებლობის მინიჭებით, კომპანიებს შეუძლიათ შეამცირონ ადმინისტრაციული და მართვის ხარჯები, რომლებიც საჭიროა საკუთარი კორპორაციული აპლიკაციების მხარდასაჭერად. SaaS ზრდის აპლიკაციების ხელმისაწვდომობას ინტერნეტში. SaaS გარანტიას იძლევა, რომ ყველა აპლიკაციის ტრანზაქცია დარეგისტრირებულია. მომხმარებელთათვის SaaS-ის უპირატესობა საკმაოდ ნათელია:

- რაციონალური მართვა;
- ავტომატიზებული განახლება და შესწორება;
- საწარმოს ჩარჩოებში მონაცემთა მთლიანობა;
- საწარმოს თანამშრომელთა ერთობლივი მუშაობა;
- გლობალური ხელმისაწვდომობა.

სერვერული ვირტუალიზაცია შეიძლება გამოყენებული იქნას SaaS არქიტექტურაში მრვალმომხმარებლიანი რეჟიმის მხარდაჭერის ან დამატების ნაცვლად. ვირტუალიზაციის პლატფორმის მთავარ უპირატესობას წარმოადგენს სისტემის პროდუქტიულობის გაზრდა დამატებით პროგრამირების აუცილებლობის გარეშე. რესურსების ერთდროული გამოყენების ეფექტი და ვირტუალიზაციის პლატფორმა SaaS-ის გადაწყვეტილებებში უზრუნველყოფს დიდ მოქნილობას და პროდუქტიულობას საბოლოო მომხმარებლისათვის.

კომუნიკაცია როგორც სერვისი (CaaS)

კომუნიკაცია როგორც სერვისი (CaaS) - ეს არის ღრუბელში საწარმოსათვის კომუნიკაციური გადაწყვეტილების აგება. ასეთი ტიპის ღრუბლოვანი გადაწყვეტილების მიმწოდებლები პასუხისმგებელი არიან იმ აპარატურის და პროგრამული უზრუნველყოფის მართვაზე, რათა უზრუნველყონ:

- საკომუნიკაციო სისტემა, რომელიც უზრუნველყოფს ინტერნეტის ან სხვა IP-ზე დაფუძნებული ქსელების (VoIP) ხმოვანი სიგნალის გადაცემას;
- მყისიერი შეტყობინების (IM) გაცვლა;
- ვიდეოკონფერენცია.

SaaS მოდელისაგან მცირედით განსხვავებულმა ამ მოდელმა თავისი ევოლუციური პროცესი დაიწყო ტელეკომუნიკაციების ინდუსტრიაში და იყო პროგრამული უზრუნველყოფის მიწოდების მომსახურების სექტორის შედეგი. SaaS ვენდორები პასუხისმგებლები არიან მათი მომხმარებლების აპარტორული და პროგრამული უზრუნველყოფის მართვაზე. SaaS ვენდორები, როგორც წესი, სთავაზობენ გარანტირებულ მომსახურების ხრისხს (QoS) მომსახურების სერვისების შეთანხმების (SLA) შესაბამისად.

CaaS-ის მოდელი საქმიან კლიენტებს საშუალებას აძლევს, მომსახურების გამოყენების დროის გადახდის საფუძველზე, შერჩევით განათავსოს კომუნიკაციისა და მომსახურებების საშუალებები. CaaS შემუშავებული ზოგადი დანიშნულების საფასო პოლიტიკაზე დაყრდნობით სთავაზობს მომხმარებლებს ყოვლისმომცველ, მოქნილ და ადვილად გასაგებ მომსახურების გეგმას.

CaaS- ის სერვისების შეთავაზება ხშირად დაკავშირებულია და მოიცავს ტრადიციული ხმის (ან VoIP) და მონაცემების ინტეგრირებას, დამატებითი ერთიანი კომუნიკაციების ფუნქციონირებას, როგორიცაა ვიდეო ზარები, თანამშრომლობა, საუბრები, რეალურ დროში ყოფნა და შეტყობინება, სატელეფონო ქსელი, ადგილობრივი და განაწილებული ხმოვანი მომსახურება, ხმოვანი ფოსტა. CaAS გადაწყვეტა მოიცავს გადაჭარბებულ გადართვას, ქსელს, აღჭურვილობის გადაჭარბებულობას, WAN *failover* - რომელიც აუცილებლად შეესაბამება მომხმარებელთა საჭიროებებს. მაღალი ხელმისაწვდომობისა და სიცოცხლისუნარიანობისათვის ყველა VoIP სატრანსპორტო კომპონენტი განლაგებულია გეოგრაფიულად გადანაწილებულ, უსაფრთხო საინფორმაციო ცენტრებში. CaaS-ი მცირე და

საშუალო ბიზნესისათვის გულისხმობს მოქნილობას და სკალარულობას, რასაც ხშირად თავად კომპანიები ვერ უზრუნველყოფენ. CaaS სერვისის პროვაიდერები მომზადებულები არიან პიკური დატვირთვისთვის, მომხმარებელის საჭიროებიდან გამომდინარე მომსახურებას წევენ მოწყობილობების მოცულობების გაზრდის, მდგომარეობის და დაფარვის ზონების გაზრდის მიზნით. ქსელის გამტარუნარიანობა და საშუალებათა ნაკრები შეიძლება შეიცვალოს დინამიურად, ამგვარად, ფუნქციონირება სამომხმარებლო მოთხოვნილებას ემთხვევა და მიმწოდებლის საკუთრებაში არსებული რესურსები უშედეგოდ არ გამოიყენება. მომსახურების მიმწოდებლისგან განსხვავებით, კლიენტის პერსპექტივას რეალურად არ მივყავართ მოძველებული აღჭურვილობის მომსახურებისკენ, ვინაიდან CaaS - ის მომსახურების მიმწოდებლის ვალდებულება იმაში მდგომარეობს, რომ პერიოდულად განაახლოს ან შეცვალოს აპარატორული და პროგრამული უზრუნველყოფა, რათა შეინარჩუნოს პლატფორმა ტექნოლოგიურად აქტუალურ მდგომარეობაში.

CaaS არ მოითხოვს კლიენტების მხრიდან კონტროლს. ეს თავიდან აცილებს კლიენტებს ინტრასტრუქტურაში რაიმე კაპიტალდაბანდებას და გამორიცხავს დამატებით ხარჯს ინფრასტრუქტურისათვის. CaaSის გადაწყვეტილებით, მომხმარებელს შეუძლია გააძლიეროს საწარმოს საკომუნიკაციო მომსახურება, საკუთარი ორგანიზაციის შიგნით გადაწყვეტილების აგების აუცილებლობის გარეშე. ეს საშუალებას აძლევს მომხმარებელს გადაანაწილოს ბიუჯეტი და პერსონალის შრომის დანახარჯი, გამოიყენოს ისინი იმ ადგილებში, სადაც ეს აუცილებელია.

VoIP კერძო საფუძველია ტელეფონის ყურმილიდან დაწყებული, რომლის ნახვაც შესაძლებელია თითოეული თანამშრომლის მაგიდაზე, თანაშრომლის ნოუთბუქზე საკლიენტო პროგრამულ უზრუნველყოფით დამთავრებული, და ამ კომპონენტებს შორის ყველა აუცილებელი მოქმედება CaaSის გადაწყვეტაში CaaS მომსახურების მიმწოდებლის მიერ მხარდაჭერილია 24/7 რეჟიმში.

- განთავსების გადაწყვეტილება და მართვა.

ინფრასტრუქტურული მომსახურების დისტანციური მართვა, რომელიც მესამე პირის მიერაა უზრუნველყოფილი კომპანიების უმრავლესობისთვის თითქოსდა მიუღებელია. თუმცა, ბოლო ათწლეულის განმავლობაში ტექნოლოგიების, ქსელისა და პროგრამული უზრუნველყოფის განვითარებით დამოკიდებულება შეიცვალა. ეს ნაწილობრივ

დაკავშირებულია შერჩეული მომსახურების გამოყენებისას ხარჯის შემცირებასთან. თუმცა, ერთეული სერვისებისგან განსხვავებით, CaaS-ის მომსახურების მიმწოდებლის შეთავაზება უზრუნველყოფს სრულ საკომუნიკაციო გადაწყვეტას, რომელსაც მთლიანად ერთი ვენდერი ახორციელებს. სხვა ფუნქციებთან ერთად, როგორიცაა VoIP და ერთიანი კომუნიკაციები, საოფისე ავტომატური სატელეფონო გაცვლის დამატებითი ფუნქციონირების ინტეგრაცია იმართება ერთი ვენდორის მიერ, რომელიც პასუხისმგებელია მთლიან ინტეგრაციაზე და მომხმარებლებისთვის მომსახურების მიწოდებაზე.

- მართვის მოხერხებულობა და ფუნქციონალურობა

როდესაც კლიენტები სარგებლობენ კავშირების მომსახურებით CaaS-ის მომსახურების მიმწოდებლის მხარეს, ისინი იხდიან მხოლოდ აუცილებელი ფუნქციონალურობისთვის. მომსახურების მიმწოდებელს შეუძლია გაანაწილოს მომსახურების ღირებულება. როგორც ადრე აღვნიშნეთ, ეს კლიენტებისათვის ზოგადად აუცილებელი ფუნქციონალურობის უფრო ეკონომიურად დანერგვის და გამოყენების საშუალებას იძლევა. პროდუქტიულობის გაზრდის ხარჯზე ეკონომია მომსახურების მიმწოდებლებს საკმაოდ მოქნილი მომსახურების საშუალებას აძლევს, ისინი არ არიან დაკავშირებულნი ინვესტიციის ერთადერთ მიმწოდებელთან. მომსახურების მიმწოდებლებს შეუძლიათ გააძლიერონ გადაწყვეტილება ანალოგიურ მიმწოდებელთა შორის საუკეთესოსთან. ისეთებთან, როგორებიცაა Microsoft, Google, Amazon, Cisco, Nortel.

- არ არის ხარჯი მოწყობილობაზე

ყველა მოწყობილობა განლაგებულია CaaS მომსახურების მიმწოდებლებთან, ეს ფაქტობრივად გამორიცხავს კლიენტებისათვის საკუთარი საინფორმაციო ცენტრების და აღჭურვილობის შესანარჩუნების საჭიროებას. არ არის ხარჯები ელექტრომოხმარებაზე, გაგრილებაზე, ფართის იჯარით აღებაზე. კლიენტი მსხვილი კომპანიის მასშტაბით მონაცემთა დამუშავების ცენტრის გამოყენებით იღებს მრავალმხრივ სარგებელს სრული რეზერვირებით - და ეს ყველაფერი შედის ყოვეთვიურ გადასახადაში.

- ბიზნესის გარანტირებული უწყვეტობა

ავარიული რეაბილიტაციის გეგმა საშუალებას მოგცემთ ავარიის შემდეგ განაგრძოთ უწყვეტი მუშაობა თქვენი ბიზნესის მონაცემთა ცენტრში? რამდენ ხანს შეძლებს თქვენი კომპანია ელექტროენერგიის გათიშვის შემდეგ მუშაობას? უმეტესობა კომპანიებისათვის ეს

მოვლენა გარდაუვალად ნიშნავს საგრძნობ ფინანსურ დანაკარგს, რომელიც დაკავშირებულია ბიზნესის გაცდენასთან. კომპანიის საინფორმაციო სისტემის განაწილება გეოგრაფიულად დაშორებულ მონაცემთა ცენტრების შორის, კომპანიათა დიდი რაოდენობისთვის ნორმად ხდება. ეს ამცირებს ფინანსური დანაკარგების რისკს და საშუალებას აძლევს იმ კომპანიებს, რომლებიც მდებარეობს იმ ადგილებში, სადაც იყო გარკვეული კატასტროფული მოვლენები, რაც შეიძლება მალე აღადგინოს ინფრასტრუქტურა. ეს პროცესი ხორციელდება CaaS-ის მომსახურების მიმწოდებლებით. ხმოვან მონაცემთა გადაცემასთან მომუშავე კომპანიების დიდი რაოდენობისათვის, სისტემის გაუმართაობა კატასტროფულია. მონაცემთა მთლიანობისგან განსხვავებით, ხმოვანი ქსელის გაუმართაობის ერთეული წერტილების აღმოფხვრა, როგორც წესი, საკმაოდ ძვირია პროექტის ფართომასშტაბიანობის და მართვის სირთულის გამო. CaaS-ის გადაწყვეტილებებს გააჩნიათ სისტემების სიჭარბის მრავალჯერადი დონეები, რაც გამორიცხავს სისტემიდან ცალკეული წერტილების გაუმართაობას.

მონიტორინგი როგორც სერვისი (MaaS)

მონიტორინგი როგორც სერვისი *(Monitoring-as-a-Service, MaaS)* წარმოადგენს ღრუბელში უსაფრთხოებით უზრუნველყოფილ მომსახურებას, უპირველეს ყოვლისა ბიზნესპლატფორმებზე. გასული ათწლეულების მანძილზე *MaaS* უფრო და უფრო პოპულარული გახდა. ღრუბლოვანი გამოთვლების გამოჩენით *MaaS* -ის პოპულარობა უფრო გაიზარდა. *უსაფრთხოების კონტროლი* ეხება კლიენტების (ბიზნესის და მთავრობის) კიბერ საფრთხეებისგან დაცვას. უსაფრთხოების სამსახური ასრულებს მნიშვნელოვან როლს კონფიდენციალობის, მთლიანობის და IT საშუალებების ხალმისაწვდომობის უზრუნველყოფასა და ხალშეწყობაში. თუმცა, დრო და შეზღუდული რესურსები კომპანიების უმეტესობისთვის ზღუდავს უსაფრთხოების ღონისძიებებს და მათ ეფექტურობას. ეს მოითხოვს ინფრასტრუქტურის და კრიტიკული ინფორმაციული საშუალებების უსაფრთხოებაზე მუდმივ ზედამხედველობას. მრავალი საწარმოო წესი მოითხოვს, რომ ორგანიზაციებმა გააკონტროლონ თავიანთი უსაფრთხოების გარემო, სერვერების ჟურნალები და სხვა საინფორმაციო საშუალებები, რათა უზრუნველყონ ამ სისტემების მთლიანობა. თუმცა ეფექტური უსაფრთხოების მონიტორინგი შეიძლება იყოს შემაშინებელი ამოცანა, რადგან ის საჭიროებს მაღალ ტექნოლოგიებს, კვალიფიციური უსაფრთხოების ექსპერტებს და

სკალარულობინ პროცესს, რომელთაგან არცერთი არ არის იაფი. უსაფრთხოების მონიტორინგის მომსახურება *MaaS* გთავაზობთ რეალურ დროში მონიტორინგს, 24/7 რეჟიმს და უსაფრთხოების ინფრასტრუქტურის მეშვეობით პრაქტიკულად მყისიერი რეაგირებას ინციდენტების მიხედვით. ეს მომსახურებები ეხმარებიან მომხმარებელთა კრიტიკული ინფორმაციის აქტივების დაცვის პროცესს. ელექტრონული უსაფრთხოების სისტემების გამოჩენამდე უსაფრთხოების მონიტორინგი და რეაგირება დიდწილად დამოკიდებული იყო ადამიანურ რესურსებსა და ადამიანურ შესაძლებლობებზე, რომლებიც მაკონტროლებელი ძალისხმევის სისწორესა და ეფექტურობას ზღუდავდნენ. ბოლო ორი ათწლეულის მანძილზე უსაფრთხოების უზრნველყოფის სისტემებში შემუშავებული იყო ინფორმაციული ტექნოლოგიები, რომლებსაც შეუძლიათ ოპერაციული უსაფრთხოების ცენტრებთან (SOC) ურთიეთმოქმედება კორპორატიული ქსელების მეშვეობით, რამაც მნიშვნელოვნად შეცვალა სურათი. ეს საშუალებები მოიცავენ ორ მნიშვნელოვან ასპექტს, ესენია:

1. საოპერაციო უსაფრთხოების ცენტრის ფლობის საერთო ღირებულება გაცილებით მაღალია, ვიდრე თანამედროვე *SOC* ტექნოლოგიების;

2. დაბალი საოპერაციო უსაფრთხოების ხარჯების მიღწევა და უსაფრთხოების საშუალებების უფრო მაღალი ეფექტურობა.

SOC-ის უსაფრთხოების მდგომარეობის კონტროლის სამსახურებს შეუძლიათ ჟურნალის და ინფრასტრუქტურული მოწყობილობებიდან დღეღამის რეალურ დროის რეჟიმში შეტყობინების აქტიურად გაანალიზების გზით გააუმჯობესონ კლიენტის უსაფრთხოების ინფრასტრუქტურის ეფექტურობა. ბრძანებების კონტროლი აკავშირებს სხვადასხვა უსაფრთხოების მოწყობილობებს იმისათვის, რომ უზრუნველყოს ანალიტიკოსები ყალბი საფრთხეების აღმოფხვრისა და რეალურ საწარმოს საფრთხეზე რეაგირებისათვის აუცილებელი უსაფრთხოების მონაცემებით. ინფორმაციული უსაფრთხოების სამსახურმა შეიძლება შეაფასოს სისტემის პროდუქტიულობა პერიოდულად განმეორებადობის საფუძველზე და, საჭიროების შემთხვევაში, გაუმჯობესების მიზნით უზრუნველყოს რეკომენდაციები.

ადრეული შეტყობინების სერვისი იტყობინება უსაფრთხოებაში ახალი სუსტი ადგილების შესახებს მისი წარმოქნისთანავე. ზოგადად, საფრთხეები დაკავშირებულია იმ წყაროებთან, რომლებსაც მესამე მხარესთან აქვთ კავშირი. ანგარიშში, ჩვეულებრივ, ელექტრონული

ფოსტით ეგზავნება კომპანიის მიერ დანიშნულ პასუხისმგებელ პირს. უსაფრთხოების სუსტ ადგლებზე ანგარიში, გარდა სუსტი ადგილების დაწვრილებით აღწერისა, ასევე მოიცავს ინფორმაციას ამ სუსტი ადგილების სისტემაზე და აპლიკაციებზე გავლენის შესახებ. უფრო ხშირად ანგარიში ასევე მიუთითებს გარკვეულ მოქმედებაზე, რომელიც უნდა შესრულდეს, რომ მოხდეს სუსტი ადგილების ეფექტის მინიმიზირება.

მართვის და მონიტორიგის სერვისის პლატფორმა ხშირად წარმოდგენილია ინსტრუმენტების პანელის სახით, რაც საშუალებას იძლევა ნებისმიერ დროს გავიგოთ სისტემის გამართულობის მდგომარეობა. შეღწევა შესაძლებლია ვებ-ინტერფეისის მეშვეობთ, რაც დისტანციურად მუშაობის საშუალებას იძლევა. თითოეული სამუშაო ელემენტი, რომელიც მოწმდება, ჩვეულებრივ შეიცავს სამუშაო ინდიკატორულ სტატუსს, ყოველთვის ითვალისწინებს თითოეული ელემენტის კრიტიკულ გავლენას. სერვისის მონაცემები საშუალებას გაძლევთ განსაზღვროთ, თუ რომელი ელემენტი იმყოფება მუშა მდგომერეობაში, რომელს არ ყოფნის სიმძლავრეები და რომელია დადგენილი პარამეტრების მიღმა. ასეთი პრობლემების იდენტიფიცირების და ამოცნობისას, შეგიძლიათ პრევენციული ღონისძიებების გატარება, რათა თავიდან იქნას აცილებული სერვისის მუშამდგომარეობის დაკარგვა.

ღრუბლოვანი გამოთვლების განლაგების მოდელების კლასიფიკაცია. როგორც ზემოთ უკვე აღვნიშნეთ, ჩვენ შეგვიძლია ღრუბლოვანი გამოთვლების სისტემების კლასიფიკაცია 4 კატეგორიაში, მათ შორის:

1. **საზოგადოებრივი ღრუბელი**. ამ მოდელში განლაგებულია მომსახურებები და ინფრასტრუქტურა, რომელიც მიეწოდება სხვადასხვა ტიპის მომხმარებელს და გამოიყენება საჯაროდ ჩვეულებრივი ადამიანების / მომხმარებლების მიერ. ამ ტიპის ღრუბელს მართავს Cloud სერვისების მომსახურების მომწოდებელი მომხმარებლებისთვის მოხმარების მიხედვით გადახდის საფუძველზე. საზოგადოებრივი ღრუბლების მაგალითებია Amazon EC2, Google App Engine და ა.შ.

2. **კერძო ღრუბელი.** ამ ტიპის ღრუბელში გამოთვლილია კომპიუტერული რესურსები ექსკლუზიურად ერთი ორგანიზაციის მიერ, რომელიც ღრუბლოვანია. ის უფრო უსაფრთხოა, ვიდრე საზოგადოებრივი ღრუბლები, რადგან მათი მომხმარებლები ორგანიზაციის შიგნით

სანდო წევრები არიან. კერძო ღრუბლების მაგალითებია IBM Cloud, Microsoft Cloud, ნებისმიერი კერძო ინსტიტუციური Cloud და ა.შ.

3 **სათემო ღრუბელი.** სათემო მოდელში ინფრასტრუქტურას რამდენიმე ორგანიზაცია ერთობლივად იყენებს იმავე პოლიტიკისა და შესაბამისობის გათვალისწინებით. ეს ხელს უწყობს ხარჯების შემდგომ შემცირებას კერძო ღრუბლებთან შედარებით, რადგან იგი ერთობლივად გამოიყენება ჯგუფების მიერ.

სხვადასხვა სახელმწიფო დონის სამთავრობო უწყებები, რომლებსაც ესაჭიროებათ გზების, საავადმყოფოების, ელექტროსადგურების ინფრასტრუქტურის შესახებ ერთიდაიგივე მონაცემების ხელმისაწვდომობა, ინფორმაციის შეგროვების მიზნით საზოგადოებრივ მოდელს იყენებენ.

4 **ჰიბრიდული ღრუბელი**. ამ განლაგების მოდელი ეხმარება ბიზნესს გამოყენოს კერძო ღრუბელში დაცული აპლიკაციებისა და მონაცემთა ჰოსტინგის უპირატესობის შესაძელებლობა, ამასთან მიიღოს ღირებულებაში სარგებელი.

ორგანიზაციამ შეიძლება შეინახოს კონფიდენციალური კლიენტური მონაცემები კერძო ღრუბლოვან აპლიკაციაში, მაგრამ ამ აპლიკაციის ჩართვა საზოგადოებრივ ღრუბელში შესაძლებელია პროგრამული უზრუნველყოფის სახით.

ცხრილი 1. ღრუბლოვანი კომპიუტერული განლაგების მოდელების შედარება

განლაგების მოდელი	მომსახურების სფერო	უზრუნველყოფილია	უსაფრთხოების დონე
საჯარო მოდელი	ზოგადი საჯარი და დიდი ინდუსტრიული ჯგუფები	ღრუბლოვანი მომსახურების მომწოდებლები	დაბალი
კერძო მოდელი	ცალკეული ორგანიზაციები	ცალკეული ორგანიზაციები	მაღალი
საზოგადოებრივი მოდელი	ორგანიზაციები, რომლებიც იზიარებენ გივე პოლიტიკას, მისიას და უსაფრთხოების ასპექტებს	რამოდენიმე ორგანიზაცია ან ღრუბლოვანი მომსახურების მომწოდებლები	მაღალი
ჰიბრიდული მოდელი	ორგანიზაციები და საჯარო	ორგანიზაციები და საჯარო	საშუალო

ღრუბლოვანი გამოთვლების ძირითადი მახასიათებლები

თუმცა არსებობს მრავალი პარამეტრი, რომლის საფუძველზეც შეგიძლიათ დაახასიათოთ ღრუბლოვანი გამოთვლების გარემო მაგრამ შესაძლებელია ღრუბელი გამოთვლითი გარემოს ხუთი მნიშვნელოვანი მახასიათებელს გამოყოფა:

1. ქსელის საშუალებით უნივერსალური შესაძლებლობების ხელმისაწვდომობა სტანდარტული მექანიზმის მეშვეობით, რომელიც ხელს უწყობს ჰეტეროგენული თხელი ან სქელი კლიენტის პლატფორმის გამოყენებას ისეთების, როგორიცაა მობილური ტელეფონები, ტაბლეტები, ლეპტოპები და ა.შ.

2. ღრუბლოვანი ინფრასტრუქტურის მასშტაბური მომსახურება ძალიან ელასტიურია კვანძებისა და მომსახურების გაფართოების მიმართ. ღრუბლების მომწოდებლებს აქვთ ღრუბლებში ახალი კვანძების და კლიენტებისათვის მომსახურებების დამატების შესაძლებლობა.

3. ღრუბლის მოთხოვნის მიხედვით თვიმომსახურება გვაძლევს საშუალებას ავტომატურად გამოვიყენოთ გამოთვლითი რესურსების შესაძლებლობები, როგორიცაა სერვერის დრო, ქსელი და საცავი ადამიანთან ურთიერთობის გარეშე.

4. გამოყენების მიხედვით გადახდის რეჟიმი, რომელსაც ღრუბლების მომსახურების მიმწოდებლების გვთავაზობს არ არის უფასო; მომხმარებელებმა უნდა გადაიხადონ მომსახურების მიღება და გამოყენებისათვის, მაგრამ მხოლოდ იმისთვის, რასაც იყენებენ.

5. Cloud თანამშრომლობა ბევრ ინდივიდუალურ ორგანიზაციას საშუალებას აძლევს, ითანამშრომლოს და იმუშაოს პრობლემის გადაწყვეტისათვის ან ნებისმიერი კვლევისთვის სხვებთან ერთად.

ღრუბელის ზოგიერთი სხვა მახასიათებელია: - საიმედოობა, მომხმარებელზე მორგება, გათვალისწინებული მომსახურება, მართვა, ვირტუალიზაცია.

Cloud სერვის ცენტრების ზოგიერთი მაგალითი

1. Google სიტყვა "ძებნის" სინონიმი გახდა. ადამიანები ხშირად ამბობდნენ, რომ "უბრალოდ დაგუგლეთ“ და თქვენ იპოვით ყველაფერს. მაგრამ ეს არ არის ერთადერთი, რასაც

Google-ის მომსახურებას გვაწვდის. ჩვენ ასევე ვიღებთ ისეთ ღრუბლოვან მომსახურებას, როგორიცაა: G-mail, Google Docs, Picasa, Google Analytics, Google Ad სიტყვები და რეკლამა.

2. Microsoft გვთავაზობს საკუთარ პლატფორმას, მომხმარებელთა და აპლიკაციის მომწოდებლებისათვის შემოთავაზებული ღრუბლოვანი სერვისების ნაკრების განსათავსებლად. სერვისები მუშაობენ Microsoft მონაცემთა ცენტრში. Microsoft- ის მიერ მოწოდებული სერვისებია: Windows Azure, SQL Azure, Windows Azure App Fabric და Windows Azure Marketplace.

3. Amazon Web Services (AWS) უზრუნველყოფს Cloud Computing პლატფორმას ყველა ზომის ბიზნესისათვის. AWS ეხმარება ბიზნეს ორგანიზაციას, საჭიროების მიხედვით, აირჩიოს საკუთარი კომპიუტერული პლატფორმა და ანაზღაუროს ის, რასაც გამოიყენებს. AWS-გამოთვლითი ღრუბლის მიერ წარმოდგენილი მომსახურებებია: გამოთვლითი ღრუბელი Amazon Elastic, Amazon Simple Storage მომსახურება, Amazon Virtual Private Cloud, Amazon Cloud front, Amazon Relational მონაცემთა ბაზა და Amazon Simple Queue მომსახურებები.

Cloud Computing- ის აპლიკაციები

Cloud Computing არის კომპიუტერული რესურსების ერთ-ერთი ყველაზე გავრცელებული სფერო, რადგან რესურსების გაზიარება და მართვა ადვილია ღრუბლის გამოყენებით. ამ თვისებებმა გახადა იგი აქტიური კომპონენტი შემდეგ სფეროებში:

1. ელექტრონული სწავლება - ეს არის ახალი ტენდენცია განათლების სფეროში, რომელიც უზრუნველყოფს სტუდენტების, ფაკულტეტის წევრებისა და მკვლევარებისთვის მიმზიდველი გარემოს შექმნას. სტუდენტებს, ფაკულტეტის წევრებს, მკვლევარებს შეუძლიათ დაუკავშირდნენ ორგანიზაციის ღრუბელს და იქიდან მიიღონ წვდომა მონაცემებზე და ინფორმაციაზე.

2. საწარმოს რესურსების დაგეგმვა (ERP). ERP- ში ღრუბლის გამოყენების საჭიროება მაშინ ჩნდება, როდესაც ნებისმიერი ორგანიზაციის საქმიანობა იზრდება. აპლიკაციების ადამიანური რესურსების, სახელფასო გადახდების და სხვ. მართვა რთული და ძვირი ხდება. იმისათვის, რომ ეს გადაილახოს, სერვისის მიმწოდებლებს შეუძლია ERP დააინსტალირონ უშუალოდ ღრუბელში.

3. ღრუბლოვანი გამოთვლების ელექტრონულმა მართვას შეუძლია გააუმჯობესოს მთავრობის ფუნქციონირება იმ საშუალებების გაუმჯობესებით რომლითაც ის უზრუნველყოფს თავიანთ მოქალაქეების მომსახურების გაწევას, ინსტიტუტებთან და სხვა მთავრობებთან თანამშრომლობას. ეს შეიძლება გაკეთდეს გარემოს ხელმისაწვდომობის გაფართოებით, რაც უფრო მეტად სკალარული და მოქნილია. მან ასევე მოეხსნება აპლიკაციების მართვის, დამონტაჟებისა და განახლების ტვირთი.

ცხრილი 2. ღრუბლოვანი გამოთვლების სხვადასხვა აპლიკაციები

აპლიკაცია	გაწეული მომსახურება
E-learning	E-mail, სიმულაციის ინსტრუმენტები, ფაილების გადაცემა, კლასის ჩაწერა, ვირტუალური საკლასო ოთახები, ვირტუალური ლაბორატორიები, გამოკითხვები, საგანმანათლებლო ფორუმები
ERP ღრუბელი	მიწოდების ჯაჭვი და ვენდორები, პროჯექთ და HR მენეჯმენტი, მომხმარებელთან ურთიერთობის მენეჯმენტი, ფინანსები და ბუღალტრული აღრიცხვა.
E-მმართველობა	საჩივრების მოგვარების სისტემა, დასაქმების მართვის სისტემა, E-პოლიცია, E- სასამართლო, გადახდა და საგადასახადო სისტემა, სოფლის მეურნეობა და საკვები, ინდუსტრია და ენერგეტიკა.

საბოლოოდ შეიძლება ითქვას, რომ ღრუბლოვანი გამოთვლები არის ფართოდ გამოიყენებადი ტექნოლოგია, რომელიც უზრუნველყოფს მომხმარებელთა მომსახურების მრავალ სახეობას მოსახურების შესაბამისი გადახდის მექანიზმის საფუძველზე. სხვადასხვა სახის ღრუბლოვანი განლაგების მოდელები ხელმისაწვდომია მომხმარებლებისათვის ინფორმაციის ხელმისაწვდომობისთვის, მაგრამ თითოეული მათგანი თავისი მნიშვნელობისაა, რაც დამოკიდებულია იმაზე, თუ როგორ გამოიყენებს მას და, შესაბამისად, განლაგების მოდელების უსაფრთხოებაც მერყეობს შესაბამისად. ცხრილი 1 აღწერს განლაგების მოდელების ფარგლებს და უსაფრთხოებას. მაგრამ კოლაბორაციული ბუნების და ღრუბლის ჰეტეროგენული გარემოს გამო უსაფრთხოება და კონფიდენციალურობის ბევრი საკითხი ღრუბლებში დომინანტური საკითხებია. ბევრი Cloud სერვისის მომწოდებელს ისეთებს, როგორიცაა Microsoft, Google და Amazon Web Services აქვს საკუთარი cloud გარემო და უზრუნველყოფს უამრავ მნიშვნელოვან Cloud მომსახურებას. Cloud Computing- ის აპლიკაციები განხილულია ცხრილში 2-ში.[14].

[14] https://www.ripublication.com/aeee_spl/aeeev4n1spl_15.pdf

თემა 9. Windows Azure SDK

ამ თავის მიზანს წარმოადგენს -Windows Azure SDK-ის შემუშვების საშუალებათა კომპლექტის გაცნობა.

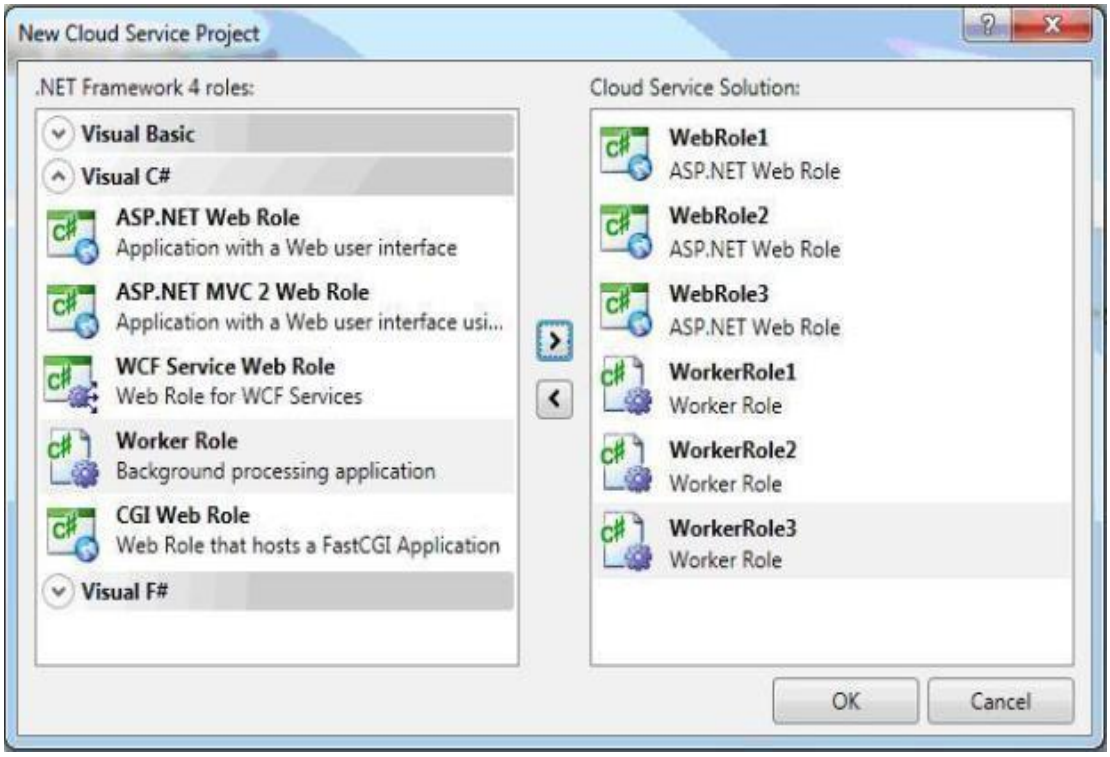
ინტერფეისპროგრამების შემმუშვებლებს Windows Azure SDK სთავაზობს აპლიკაციებს, რომლებიც აუცილებელია *Windows Azure*-ში სკალირებული მომსახურებების შემუშავების, განლაგებისა და მართვისათვის.

Azure *Cloud Fabric* და Azure მომსახურებები არ უჭერენ მხარს ღრუბელში შემუშავების ან გამართვის *ოპერაციებს*, ამიტომ *Azure SDK* უფლებას იძლევა ეს გაკეთდეს ლოკალურად *Development Fabric* (DF) და *Development Storage* (DS აპლიკაციების სახით, რომლებიც იყენებენ *Windows Azure SDK-ს.* აპლიკაციების პროგრამირების ხელშეწყობის მიზნით SDK-თან ერთად, ასევე დამონტაჟებულია აპლიკაციების მაგალითების კოლექცია და აპლიკაციების შედგენის გამარტივებისათვის შედგენილია კლასების ბიბლიოთეკები.

საჭიროა დამონტაჟდეს *NET* Framework 3.5 SP1 და *SQL* Express, ასევე აუცილებელია ჩაირთოს *ASP.NET* და WCF *HTTP* Activation *IIS* 7.0-სთვის, *Windows Server* 2008-სთვის, *Windows Vista* SP2, ან *Windows* 10 RC ან უფრო გვიან *SDK* დამონტაჟებისა და გაშვებისათვის. გამოშვების მიმართ შენიშვნები მოიცავს ამ პარამეტრების კონფიგურაციისთვის ინსტრუქციებს. SDK-ის გამოყენება არ არის აუცილებელი, რადგან შესაძლებელია ნებისმიერი ოპერაციული სისტემისა და პროგრამირების ენების გამოყენება, რომლებიც მხარს უჭერენ HTTP მოთხოვნებს და პასუხებს. თუმცა, დაინახავთ, რომ SDK- ის .NET- ის პროგრამირების ინტერფეისების და ბიბლიოთეკების გამოყენება აპლიკაციებისა და საცავებისთვის უშუალოდ HTTP-სთან შედარებით მარტივად მუშაობის საშუალებას გაძლევთ.

იმის შემდეგ, რაც დაამონტაჟეთ Azure SDK, თქვენ უნდა ჩამოტვირთოთ და დააყენოთ Windows Azure ინსტრუმენტები Visual Studio-სთვის, რათა დაამატოთ ვებ აპლიკაციის შაბლონები Sevice, Worker Cloud Service, Web და Worker Cloud Service Workflow Service. თქვენ შეგიძლიათ ჩამოტვირთოთ Windows Azure SDK- ის და Windows Azure Tools-ს ვიზუალური სტუდიის Windows-ის Azure-ის საწყისი გვერდიდან www.microsoft.com/azure/windowsazure.mspx.Visual Studio-ში Windows Azure Tools-ის ინსტალაციის შემდეგ Cloud Service თარგები გამოჩნდება დიალოგში ახალი პროექტის შესაქმნელად. Cloud Service კვანძის შერჩევისას, ახალი Cloud Service იხსნება, რომელიც

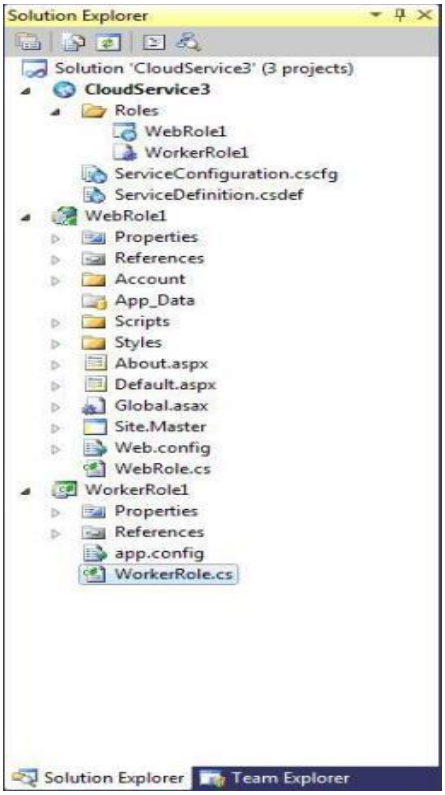
საშუალებას გაძლევთ დაამატოთ ASP.NET *Web Roles, Worker Roles* ან *CGI Web Roles* ახალი პროექტისთვის. Windows Azure SDK საშუალებას გაძლევთ დაამატოთ ერთზე მეტი როლი თითოეული ტიპის Cloud სერვისისთვის. თითოეული როლი იყენებს Windows Azure CPU- ს ცალკე ინსტანციას, ამიტომ ღრუბელში პროექტის გაშვების მინიმალური ღირებულება დაახლოებით 4 * $ 0.12 = $ 0.48 საათში იქნება.



ნახ. 25. ახალი Cloud Service პროექტის შექმნა Visual Studio- ში.

მითითებული როლების დამატებით და OK-ზე დაჭერით, გაიხსნება ახალი გადაწყვეტა *WebRole* და *Worker-Role* პროექტებით *Solution Explorer* -ში, როგორც ეს ნახ. 26-ზეა ნაჩვენები.

Roles კვანძი შეიცავს WebRole ელემენტებს, რომლებიც მიუთითებენ თითოეულ WebRole-ზე, რომელიც უზრუნველყოფს სამომხმარებლო ASP.NET ინტერფეის აპლიკაციებისათვის და თითოეულ WorkerRole-ს ის ოპერაციების გამოსათვლელად, რომლებიც არ საჭიროებენ სამომხმარებლო ინტერფეისს ან იყენებენ ASP.NET WebRole გვერდებს ამის ნაცვლად.



ნახ. 26. Solution Explorer Cloud Service.

Cloud Service-ის ტიპის მიხედვით, პროექტები მოიცავს Microsoft.ServiceHostingServiceRuntime სახელთა სივრცეს, რომელიც შეიცავს ცხრილში მოცემულ კლასებს.

კლასი	აღწერა
RoleEntryPoint	უზრუნველყოფს მეთოდებს მართვის ინიცირებისათვის, დაწყებისა და შეჩერების მომსახურების მეთოდებისათვის, ასევე გამოიყენება სტატუსის მონიტორინგის მომსახურებისათვის.
RoleException	იძლევა შეტყობნებს, როდესაც როლის შიგნით მიმდინარეობს დაუშვებელი ოპერაცია
RoleManager	უზრუნველყოფს გზავნილებისა და შემომავალი შეტყობინებების შესვლის მეთოდებს, იღებს მომსახურების კონფიგურაციის პარამეტრებს და აბრუნებს რესურსის ადგილმდებარეობას
RoleStatus	იძლევა ინფორმაციას როლის ამჟამინდელი სტატუსის შესახებ: : Healthy, NonExistent, Started, Starting, Stopped, Stopping ან Unhealthy

პროექტები, რომლებიც იყენებენ WebRole-ს შაბლონს, განსაზღვრავს ASP.NET Default.aspx ვებ-გვერდს, როგორც ამოსავალი წერტილს სამომხმარებლო ინტერფეისის აპლიკაციისათვის ღრუბელში.

ეს სერვისი აერთიანებს HelloFabric ნიმუშის აპლიკაციისგან *Common* კლასის ბიბლიოთეკას, რათა დაეხმაროს აპლიკაციის ჟურნალირების პრობლემებს. აპლიკაციების ჟურნალები არის Cloud Fabric-ში გაშვებული აპლიკაციების ჩაყრის პრაქტიკული ინსტრუმენტი. ჟურნალების წაკითხვისთვის, თქვენ უნდა გადააკოპიროთ ისინი *Blob* მასივში, პორტალის ინსტრუმენტარიების საშუალებით.

StorageClient პროექტის ნიმუში მოიცავს StorageClient-ის კლასის ბიბლიოთეკას, რომელიც .NET *Client* ბიბლიოთეკასთან ერთად, უზრუნველყოფს ADO.NET-ის მონაცემთა სერვისისთვის Microsoft-ის NET ინტერფეისის კლასს HTTP ოპერაციებისათვის Azure Blob, Queue და Table Storage Services სერვისებზე. ეს პროექტი ასევე მოიცავს საკონსოლო აპლიკაციას, რომელიც საშუალებას გაძლევთ შეამოწმოთ ბიბლიოთეკის შესაძლებლობები. საკონსოლო აპლიკაცია C # უშვებს *Development Fabric*-ში *Development Storage*-ით.

Windows Azure SDK-ის ინსტალაციისას არ არის დამონტაჟებული აპლიკაციების ის ნიმუშები, რომლებიც შედის *Program Files*|Microsoft Windows Azure SDK |1.0 |samples.zip-ში. გამოარქივებით დააინსტალირეთ ნიმუშები unzipping samples.zip დირექტორიაში, სადაც თქვენ გაქვთ *ჩაწერაზე ნებართვა*.

CloudDrive-ის მაგალითის გასაშვებად საჭიროა PowerShell.

დირექტორია, რომელშიც ამოღებულია *samples.zip* -ის არქივის შედგენილობა, ასევე შიცავს შემდეგ სამ პაკეტურ ფაილს (cmd), რომელიც შეიძლება გაიშვას ბრძანებათა სტრიქონიდან:

- *buildall.cmd* აგებს პროექტების ყველა ნიმუშს Visual Studio:-ს გამოყენების გარეშე;
- *createtables.cmd* იძახებს *buildall.cmd* -ს და ქმნის მნაცემთა ბაზას და ნიმუშებისათვის, რომელებიც იყენებენ Table Storage, საჭირო ცხრილებს;
- *rundevstore.cmd* იძახებს *createtables.cmd* -ს და უშვებს საცავის შემუშავებას, ათავსებს რა მას *createtables.cmd*.-ს მიერ შექმნილ ბაზაში.

Development Fabric -ის შემაგენლობაში შედის შემდეგი შესრულებადი ფაილები: DFAgent.exe, DFLoadBalancer.exe, DFMonitor.exe და DFService.exe, რომელიც ნაგულისხმევად დამონტაჟებულია Azure SDK ინსტალატორის მიერ \ Program Files \ Windows Azure SDK \ v1.0 \ bin \ devfabric- ში. *Development Fabric* -ის ამოცანათა მენეჯერში გაშვების შემდეგ, შეგიძლიათ იხილოთ ეს ოთხი პროცესი. ამის გაკეთება შეგიძლიათ შემდეგნაირად:

- *Development Fabric* -ის მოსახურების და მისი ინტეტფეის *DFUI.exe*-ის გასაშვებად აირჩიეთ პროგრამები *Windows Azure SDK\Development Fabric*
- შეტყობინებების პანელში Development Fabric-ის ნიშნაკზე მაუსის მარჯვენა ღილაკზე დაჭერით ავირჩიოთ Development Fabric მომსახურების გაშვება (ნახ. 5.3).
- დავაკომპილიროთ და გავუშვათ Azureაპლიკაცია Visual Studio-ში.

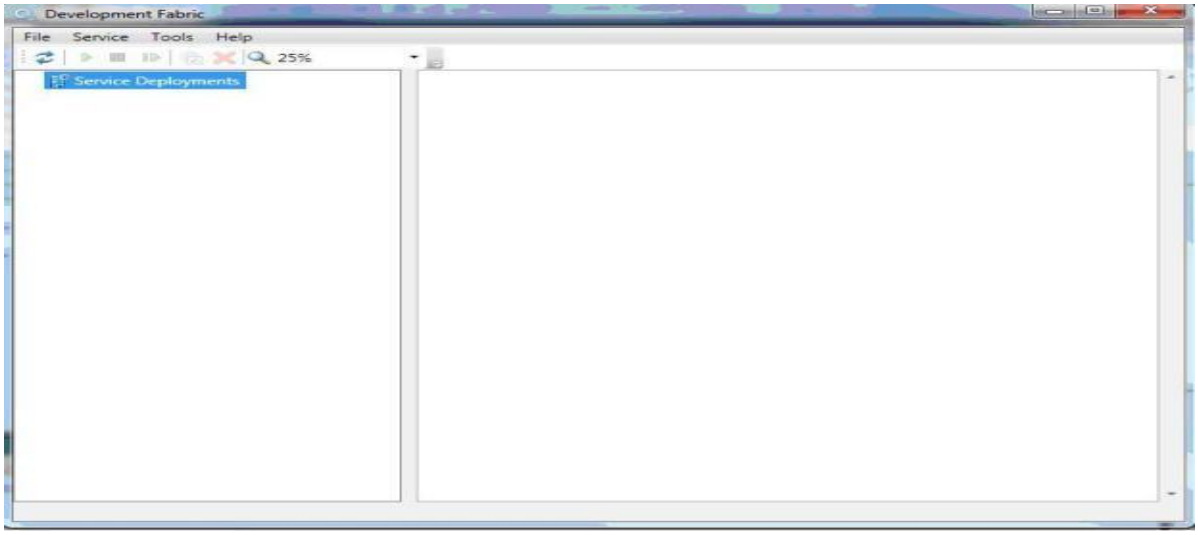
ნახ. 26. შეტყობინებების პანელში Development Fabric-ის ნიშნაკზე მაუსის მარჯვენა ღილაკზე დაჭერით მიღებული გამოსახულება.

ნახ. 27 გვიჩვენებს მომხმარებლის ინტერფეის DFUI-ს. როდესაც თქვენ უშვებთ ან აჩერებთ გამართვის შესაბამისი აპლიკაციები გამოჩნდება ან გაქრება DFUI სამომხმარებლო ინტერფეისდან.

Windows Azure პლატფორმა მხარს უჭერს სამი ტიპის სკალურობის მქონე საცავს:

- არასტრუქტურირებადი მონაცემები (blob),
- სტრუქტურირებადი მონაცემები (ცხრილები),
- კომუნიკაცია პროგრამებსა და სერვისებს შორის (რიგები).
- rundevstore.exe-ის გაშვებით ან *Visual Studio*-ში სამომხმარებლო კოდ Azure-ის აკრეფით და გაშვებით გაიშვება სამივე სერვერი, მაშინაც კი, თუ თქვენს პროექტს მხოლოდ ერთი სერვისი სჭირდება, და აისახება სამომხმარებლო ინტერფეის *Development Storage*-ში.

მონაცემების დაკარგვისგან დასაცვად, Azure ღრუბელი ბლობებს (blob), ცხრილებს და რიგებს ინახავს მინიმუმ სამ ცალკეულ კონტეინერში, ერთ მონაცემთა დამუშავების ცენტრში. Azure გეოლოკაციის ინსტრუმენტი საშუალებას მოგცემთ, კატასტროფების შემდეგ და კონკრეტული გეოგრაფიული რეგიონების მუშაობის გასაუმჯობესებლად, Microsoft მონაცემთა დამუშავების რამდენიმე ცენტრში მოახდინოთ მონაცემების დუბლირება.



ნახ. 27. Development Fabric აპლიკაციის სამომხმარებლო ინტერფეისი.

აპლიკაციებს Azure, რომელებსაც თქვენ უშვებთ *Development* Framework-ში, შეიძლება ჰქონდეთ *Development Storage* -ში ლოკალურ მონაცემებთან წვდომა ან მონაცემებთან რომელებიც ჩატვირთულია Azure ღრუბელში. აპლიკაცია მიმართავს პროექტ *Service*Configuration.cscfg-ის კონფიგურაციის ფაილში გარკვეულ ადგილას მდებარე კონკრეტულ პორტსა და მონაცემებს.

Azure ServiceDefinition.csdef პროექტის კონფიგურაციის ფაილი განსაზღვრავს სტანდარტულ შესვლის წერტილებს და კონფიგურაციის პარამეტრებს, რომლებიც ინახება ServiceConfiguration.cscfg ფაილში. ნახ. 25-ზე ნაჩვენებია ServiceDefinition.csdef ფაილის შედგენილობა ნაგულისხმევად, როდესაც შაბლონებიდან *Windows* Azure *Tools Visual Studio* -სთვის ერთ-ერთის გამოყენებით (აღნიშნულია ძირითადი მნიშვნელობები) ადგენთ Azure პროექტს.

```
<ServiceDefinition name="SampleWebCloudService"
 xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
<WebRole name="WebRole">
<InputEndpoints>
<!-- Must use port 80 for http and port 443 for https
 when running in the cloud -->
<InputEndpoint name="HttpIn" protocol="http" port="80" />
</InputEndpoints>
<ConfigurationSettings>
<Setting name="AccountName"/>
<Setting name="AccountSharedKey"/>
<Setting name="BlobStorageEndpoint"/>
<Setting name="QueueStorageEndpoint"/>
```

```
<Setting name="TableStorageEndpoint"/>
</ConfigurationSettings>
</WebRole>
</ServiceDefinition>
```

ლისტინგი 1. ServiceDefinition.csdef ფაილის შედგენილობა

InputEndpoint-ის მნიშვნელობა მხოლოდ ღრუბლებში შენახვისათვის გამოიყენება.

ამონაბეჭდი 2-ში გვჩვენებს ServiceDefinition.csdef ფაილის შედგენილობას ვებ-აპლიკაციისათვის *SampleWebCloudService* ნაგულისხმევად კონფიგურაციით *Development Storage* -სთვის (გამოყოფილია):

```
<?xml version="1.0"?>
<ServiceConfiguration serviceName="SampleWebCloudService"
 xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration">
<Role name="WebRole">
<Instances count="1"/>
<ConfigurationSettings><Setting name="AccountName" value="devstoreaccount1"/>
<Setting name="AccountSharedKey" value="Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ
 1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw=="/>
<Setting name="BlobStorageEndpoint" value="http://127.0.0.1:10000/"/>
<Setting name="QueueStorageEndpoint" value="http://127.0.0.1:10001/"/>
<Setting name="TableStorageEndpoint" value="http://127.0.0.1:10002/"/>
<!--<Setting name="AccountName" value="oakleaf"/>
<Setting name="AccountSharedKey" value="3elV1ndd . . . Coc0AMQA==" />
<Setting name="BlobStorageEndpoint"
 value="http://blob.core.windows.net" />
<Setting name="QueueStorageEndpoint"
 value="http://queue.core.windows.net" />
<Setting name="TableStorageEndpoint"
 value="http://table.core.windows.net" />
</ConfigurationSettings>
</Role>
</ServiceConfiguration>
```

ლისტინგი 1. ServiceConfiguration.cscfg ფაილის შედგენილობა

ServiceConfiguration.csfg-ის ფაილის ელემენტების აღწერა:

- Instances count - თქვენი აპლკაციის ეგზემპლიარების რაოდენობა, რომელიც შეიქმნება ღრუბლებში, როდესაც განათავსებთ მას.
- AccountName - თქვენი *Hosted Service*-ით შექმნილი ასოცირებული სახელი, რომელითაც თქვენ შექმენით სააღრიცხვო ჩანაწერი Development Storage -სთვის არის devstoreaccount1.
- AccountSharedKey ახდენს რამდენიმე ელემენტის დაშიფრას HTTP მოთხოვნაში.

- BlobStorageEndpoint არის საზოგადოებრივი მუდმივი Universal Resource Identifier (URI). Developer Storage -სთვის ეს არის loopback კომპიუტერის ინტერფეისის მისამართი (ლოკჰოლოთი = 127.0.0.1) ნაგულისხმევი TCP პორტით 10000.
- QueueStorageEndpoint Cloud- ში შენახვისთვის ეს არის საზოგადოებრივი მუდმივი URI. Developer Storage ეს არის loopback კომპიუტერის ინტერფეისის მისამართი, რომელსაც გააჩნია ნაგულისხმევი 10001 TCP პორტი.
- TableStorageEndpoint ეს არის საზოგადოებრივი მუდმივი URI. Developer Storage -სთვის loopback კომპიუტერის ინტერფეისის მისამართი, რომელსაც გააჩნია ნაგულისხმევი 10002 TCP პორტი.

შეგიძლიათ შექმნათ თქვენი საკუთარი *TCP*-ის ნომრები თუ ნაგულისხმევ მნიშვნელობათა გამოყენებისას წარმოიშვება კონფლიქტში არსებულ კონფიგურაციასთან.

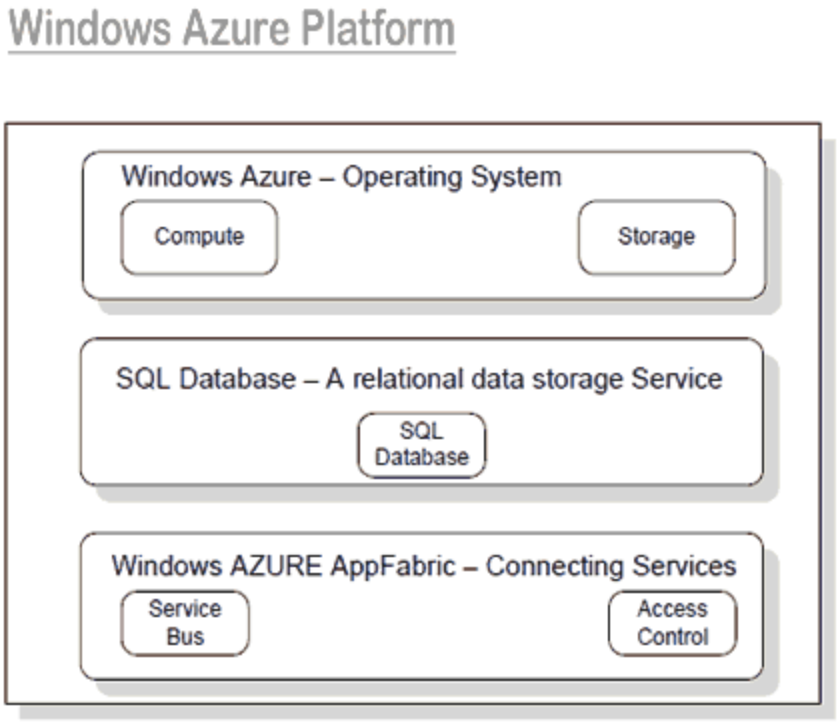
თემა 10. Azure Service Platform

არქიტექტურა Windows Azure Platform

Windows Azure პლატფორმა - ეს არის პლატფორმა როგორც სერვისის მოდელი, რომელიც გულისხმობს აპლიკაციბის *გაშვებას* სერვერებზე და მასთან დაკავშირებული იმ ქსელური ინფრასტრუქტურებზე, რომელებიც მდებარეობენ Microsoft მონაცემთა ცენტრებში და ხელმისაწვდომია *ინტერნეტით*. Azure პირველად გამოცხადდა 2008 წლის ოქტომბერში, კოდური სახელით „პროექტის წითელი ძაღლი“ და გამოიშვა 2010 წლის 1 თებერვალს, როგორც „Windows Azure“ 2014 წლის 25 მარტს სახელის "Microsoft Azure" შეცვლამდე. პლატფორმა შედგება სკალირებადი „ღრუბლოვანი“ ოპერაციული სისტემისაგან, მონაცემთა შენახვის ფაბრიკისაგან და, ფიზიკური ან ლოგიკური (*ვირტუალური*) *Windows Server* 2008-ის ეგზემპლიარების გავლით მიწოდების, მაკავშირებელი სერვერებისგან. Microsoft Azure განვითარების ინსტრუმენტების ნაკრები (SDK) უზრუნველყოფს „ღრუბლოვანი სერვისების“ ვერსიების განვითარებას, ისევე კარგად, როგორც Windows Azure-ში სკალირებადი სერვისების განვითარების, განლაგების და მართვისთვის აუცილებელი გამოყენებითი პროგრამირების (*API*) ინსტრუმენტები და ინტერფეისები, *Visual Studio* 2008 და 2010-თვის Azure-ის აპლიკაციების შაბლონების ჩათვლით. ნახ 28-ზე გამოსახულია "ღრუბელოვან" პლატფორმის ბაზაზე კომპონენტები და დეველოპერის კომპონენტები.

Microsoft-ის თანახმად, Azure-ის გამოყენებისას თქვენ იღებთ:

- ვებ-სერვისებთან მუშაობისათვის არსებული აპლიკაციების ადაპტაციას;
- ინტერნეტში აპლიკაციების აგებას, მოდიფიკაციას და განაწილებას მინიმალური ადგილობრივი რესურსებით;
- ასრულებთ ისეთი სერვისებს, როგორიცაა მონაცემთა დიდი ოდენობის შენახვა, მონაცემების პაკეტური დამუშავება, მონაცემთა დიდი რაოდენობით გამოთვლა და ა.შ.
- ვებ სერვისების შექმნას, ტესტირებას, გამართვას და გავრცელებას სწრაფად და ეფექტურად;
- ადგილობრივი რესურსების აგების და დისტრიბუციის ხარჯებისა და რისკების შემცირება;
- IT მენეჯმენტის ხარჯებისა და ძალისხმევის შემცირება.



[15]

ნახ. 28. Windows Azure პლატფორმის კომპონენტები და განვითარების ინსტრუმენტების კომპლექტი.

Microsoft-ის ეკონომიკურ მდგომარეობა Azure-ის პლატფორმის გამოშვების დროს აქცენტირებას ახდენდა ხარჯების შემცირებაზე, რაც მცირე და მსხვილი IT განყოფილებების მიერ Azure-ს გამოყენების ძირითად მიზეზს წარმოადგენს.

Microsoft-მა შეიმუშავა Azure პლატფორმა, რომელიც საშუალებას აძლევს .NET- ის დეველოპერებს გააუმჯობესონ Visual Studio 2008-ში (და ზემოთ) ASP.NET ვებ პროგრამებისა და Windows Communication Framework (WCF) სერვისების შექმნის გამოცდილება. ვებ-აპლიკაციებს პროექტები გაიშვებიან ინტერნეტის საინფორმაციო სერვისების იზოლირებულ ვერსიაში (IIS) 7. ვებ-აპლიკაციების და ვებ-სერვისების გაშვება ნაწილობრივ სანდო უსაფრთხოების მექანიზმში ხდება, რომელიც საშუალებას გაძლევთ შეზღუდოთ კომპიუტერულ რესურსებზე კოდის *ხელმისაწვდომობა* (Code Access *Security)*, რომელიც დაახლოებით შეესაბამება ASP.NET ნდობის საშუალო დონეს და ზღუდავს ოპერაციული სისტემის გარკვეული რესურსებზე ხელმისაწვდომობას. Windows Azure შემუშავების ნაკრები (მარტი 2009) საშუალებას იძლევა .NET კოდის გასაშვებად გამოიყენებული იყოს ზოგიერთ ოპერაციულ სისტემებთან სრული ხელმისაწვდომობა, გამოიყენებული იყოს NET

[15] https://vkinfotek.com/azureqa/what-are-the-components-of-windows-azure-platform.html

ბიბლიოთეკები, რომლებიც საჭიროებენ სრულ ნდობას და, პროგრამული არხების (*Pipe*) გამოყენებით, ურთიერთქმედების პროცესის დამუშავებას. Microsoft *Ruby*, *PHP* და Python პროგრამული კოდის „ღრუბლოვან“ პლატფორმაში გაშვების მხარდაჭერას ჰპირდება. შემუშავების პლატფორმის საწყისი ვერსია შეზღუდული იყო Visual Studio2008 და უფრო ზევით პროგრამირების გარემოთი Eclipse-ის ხელსაწყოების გეგმის მხარდასაჭერად. Windows Azure პლატფორმა მხარს უჭერს ვებ სტანდარტებს და ოქმებს, მათ შორის SOAP, HTTP, XML, Atom და AtomPub.

შემუშავებებისათვის ღრუბელში ASP.NET აპლიკაციის განთავსების შესვლის საწყის წერტილს წარმოადგენს Windows Azure პორტალი შემდეგ მისამართზე https://windows.azure.com/Cloud/Provisioning/Default.aspx. პორტალი მოითხოვს შესვლას *Windows Live ID*-ის გამოყენებით. Azure-ის წინასწარი ვერსია (Community *Technical* Previews, *CTPs*) საჭიროებს სხვადასხვა ტოკენს შემდეგისათვის:

Windows Azure-სთვის, რომელიც მოიცავს:

- Azure Hosted service;
- Storage Accounts.

SQL Azure *Live* მომსახურებისთვის, რომელიც მოიცავს:

- Live Framework: წინასწარი ვერსია;
- Live Services: გამოყენებითი პროგრამირების არსებული ინტერფეისი.

Live Services: გამოყენებითი პროგრამირების არსებული ინტერფეისები არ არიან წინასწარი ვერსიის ნაწილები და არ საჭიროებენ ტოკენებს. 2009 წლის დასაწყისიდან, Windows Azure ტოკენი Hosted *Service* ერთ სააღრიცხვო ჩანაწერის, *Storage* -სთვის ორი სააღრიცხვო ჩანაწერის უფლებას იძლევა. თქვენ ითხოვთ ტოკენს Azure სიმბოლოს მეშვეობით Microsoft Connect გვერდს, რომელზედაც შეგიძლიათ მოხვდეთ პორტელის გვერდიდან.

2009 წლის მარტის წინასწარი ვარიანტი მოიცავს გეოლოკაციას, რომელიც საშუალებას აძლევს სააღრიცხვო ჩანაწერის მფლობელებს შეარჩიონ მონაცემთა ცენტრები *Hosted Services* -ის და *Storage* Accounts-ის განლაგებისათვის. მაგალითად, USA-Northwest (Quincy, WA) და USA-Southeast (*San* Antonio, TX.). თქვენ შეგიძლიათ ჯგუფში დაამატოთ *Hosted Services* და *Storage* Accounts-ის ნაკრებები იმისათვის, რომ მომსახურება და საცავები

გარანტირებულად ერთიდაიმავე მონაცემთა დამუშვების ცენტრში არის განლაგებული, რათა გავზარდოთ *პროდუქტიულობა*.

თქვენ შეგიძლიათ ჩადოთ GUID გაგზავნა მიღებული Azure გუნდის წევრისაგან ტექსტურ ფანჯარაში და დააწვეთ Claim *Token* -ს იმისათვის, რომ დაამატოთ შემავალი ობიექტისათვის კიდევ ერთი ან მეტი წერტილი ან სიაში რამდენიმე ობიექტი.

თუ გსურთ ისეთი ვებ-გვერდების შემუშავება, რომელებსაც მხარს უჭერენ Live Framework ან Mesh (მონაცემთა სინქრონიზაციის პროგრამული პაკეტი Microsoft-ის მიერ შექმნილ ჯვარედინი პლატფორმის გარემოში), თქვენ უნდა მოითხოვოთ *ტოკენ Live* Framework *email*meshctpe@*microsoft.com*.-ის მეშვეობით. *ტოკენ Live* Framework-ს მიღებისთანავე შეგიძლიათ ჩამოტვირთოთ და დააყენოთ *ტოკენ Live* Framework SDK-ისა და *Live* Framework *Tools Visual Studio* -სთვის მიმდინარე ვერსიები გვერდზე მითითებულ ბმულზე http://dev.live.com/liveframework/sdk/. თქვენ უნდა გადაიხადოთ LiveFramework ტოკენში, რათა ჩამოტვირთოთ პირდაპირი SDK და დამატებითი ინსტრუმენტები. არ არსებობს *Windows*Azure სააღრიცხვო ჩანაწერის გამოყენების აუცილებლობა Azure *Hosted Services and Storage* Services,-ის ტესტირებისათვის, რადგან შემუშავების პლატფორმა Azure „ღრუბლოვანი“ სერვისების ემულაციას ახდენს თქვენს კომპიუტერში.

Windows Azure Storage

Windows Azure Storage საცავი უზრუნველყოფს შემმუშავებლებისათვის ღრუბელში მონაცემების შენახვის შესაძლებლობას. აპლიკაციას შეუძლია თავის მონაცემებთან წვდომა ნებისმიერ დროს პლანეტის ნებისმიერ წერტილიდან განახორციელოს, შეინახოს ნებისმიერი მოცულობის მონაცემი საკმაოდ დიდხანს. ამასთან მონცემები გარანტირებულად არ იქნება დაზიანებული და დაკარგული. Windows Azure Storage გვთავაზობს მონაცემთა აბსტრაქციის მდიდარ ნაკრებს:

- Windows Azure Table - სერვისის მდგომარეობების საცავების სტრუქტურირებას.
- Windows Azure Blob - უზრუნველყოფს მონაცემთა დიდი ელემენტების საცავებს.
- Windows Azure Queue - უზრუნველყოფს სერვისების შორის მონაცემთა გაცვლის რეალიზაციის ასინქრონული ამოცანების გადაგზავნას.

Azure Table Services

Azure Table Services -ეს არის სტრუქტურირებული საცავი, რომელიც მხარს უჭერს მაღალ საკალარულობის ცხრილებს ღრუბელში, რომელიც შეიძლება შეიცავდეს მილიარდ პირს და ტერაბაიტის მონაცემებს. ტრაფიკის გაზრდასთან ერთად, სისტემა ათასობით სერვერის ავტომატურად დაკავშირებით ეფექტურად სკალირდება. სტრუქტურირებული საცავი ხორციელდება ცხრილების სახით (Tables), რომელშიც არის ობიექტები (Entities), რომლებიც შეიცავენ გარკვეულ დასახელებულ თვისებებს (Properties). Windows Azure Table -ის რამდენიმე ძირითადი მახასიათებელია:

- LINQ, ADO. NET Data Services და RES-ის მხარდაჭერა.
- ADO .NET Data Services კლიენტის ბიბლიოთეკის გამოყენებისას კომპილაციის დროს ტიპების კონტროლი.
- მონაცემთა ტიპების მდიდარი ნაკრები მახასიათებლების მნიშვნელობებისათვის.
- შეზღუდავი რაოდენობის ცხრილების მხარდაჭერა და ობიექტები ცხრილების ზომის შეზღუდვის გარეშე.
- თითოეული ობიექტისათვის მთლიანობის მხარდაჭერა.
- განახლებისა და წაშლის არამყარი ბლოკირება.
- მოთხოვნებისათვის, რომლებიც საჭიროებენ დროის ხანგრძლივ პერიოდს ან მოთხოვნებისათვის, რომლებიც შეწყვეტილია დალოდების დროის დამთავრების გამო, აბრუნებენ ნაწილობრივ შედეგებს, და გაგრძელების მარკერს.

განვიხილოთ Windows Azure Table: ცხრილის მონაცემთა მოდელი.

- **საცავის სააღრიცხვო ჩანაწერი (Storage Account)** - აპლიკაცია Windows Azure Storage-სთან შესაღწევად უნდა გამოვიყენოთ ნამდვილი სააღრიცხვო ჩანაწერი. ახალი სააღრიცხვო ჩანაწერი შეიძლება შევქმნათ Windows Azure პორტალის ვებ-ინტერფეისის გამოყენებით. როგორც კი სააღრიცხვო ჩანაწერი შეიქმნება, მომხმარებელი იღებს 256-თანრიგიან სადუმლო გასაღებს, რომელიც შემდგომში გამოიყენება საცავის სისტემებში ამ მომხმარებლის მოთხოვნათა აუთენტიფიკაციისათვის. კერძოდ, ამ საიდუმლო გასაღების მეშვეობით იქმნება მოთხოვნისათვის ხელმოწერა HMAC SHA256. ეს ხელმოწერა გადაეცემა ამ მომხმარებლის თითოეულ მოთხოვნასთან ერთად აუთენტიფიკაციის უზრუნველსაყოფად. სააღრიცხვო ჩანაწერის სახელი შედის URL-ში ჰოსტის სახელების შემადგენლობაში. ცხრილებთან

შესაღწევად გამოიყენება ჰოსტის სახელის შემდეგი ფორმატი: <სააღრიცხვო ჩანაწერის სახელი >.table.core.windows.net.

- **ცხრილი (Table)** – შეიცავს ობიექტების ნაკრებს. ცხრილების სახელების მოქმედების არე შემოსაზღვრულია სააღრიცხვო ჩანაწერით. აპლიკაციამ შეიძლება საცავის სააღრიცხვო ჩანაწერის ჩარჩოებში შექმნას მრავალი ცხრილი.
- **ობიექტი (სტრიქონი) (Property (Column))** - ობიექტები (ობიექტი წარმოადგენს „სტრიქონის“ ანალოგს) - ეს არის მონაცემთა ძირითადი ელემენტები, რომლებიც ცხრილში ინახება. ობიექტი მოიცავს მახასიათებლების ნაკრებს. თითოეულ ცხრილს გააჩნია ორი მახასიათებელი, რომლებიც ქმნიან ობიექტისათვის უნიკალურ გასღებს.
- **მახასიათებელი (სვეტი) (Property (Column))** - წამოადგენს ობიექტის ცალკეულ მნიშვნელობებს. მახასითებლების სახელები რეგისტრის მიმართ არიან მგრძნობიარეები.
- **სექციის გასღები (RowKey)** - ეს არის თითოეული ცხრილის პირველი მახასითებებლი. ეს სისტემა იყენებს მოცემულ გასაღებს ცხრილების მახასიათლებლების ავტომტური განაწილებისათვის, შენახვის მრავალ კვანძს შორის.
- **დროის ნიშნული (Timestamp)** - ყოველ ობიექტს გააჩნია ვერსია, რომელსაც ინახავს სისტმა.
- **სექცია (Partition)** - ეს არის ცხრილში ერთნაირი სექციის გასაღების მნიშვნელობის მქონე ობიექტების ნაკრები.
- **დახარისხების წესრიგი (Sort Order)** - CTP ვერსიისთვის მხოლოდ ერთი ინდექსია გათვალისწინებული, რომელშიც ყველა ობიექტი დაყოფილია PartitionKey-ად და შემდეგ RowKey-ად.

ცხრილს გააჩნია მოქნილი სქემა. Windows Azure Table თვალყურს ადევნებს თითოეული ობიექტის თითოეული მახასიათებლის სახელს და ტიპიზირებულ მნიშვნელობას. აპლიკაციას შეუძლია კლიენტის მხარეს ფიქსირებული სქემის მოდელირება, უზრუნველყოფს რა ყველა შექმნილი ობიექტისათვის მახასიათებლების ერთნაირ ნაკრებს.

განვიხილოთ რამდენიმე დამატებითი ინფორმაცია ობიექტების შესახებ:

- ობიექტს შეიძლება ჰქონდეს 255-მდე მახასიათებელი, აუცილებელი სისტემური მახასიათებლია ჩათვლით: PartitionKey, RowKey და Timestamp. ობექტების დანარჩენი მახასიათებლის სახელები განისაზღვრება აპლიკაციების მეშვეობით.

- სტრიქონური ტიპის მახასიათებელები PartitionKey, RowKey და Timestamp.
- Timestamp მახასიათებელი ხელმისაწვდომია მხოლოდ მახასიათებლების მომსახურე სისტემით წაკითხვისათვის, რომელიც უნდა განხილოს როგორც გაუმჭვირვალე მახასიათებელი.
- ფიქსირებული სქემის არარსებობა - Windows Azure Table არ ინახავს არანაირ სქემას, ამიტომ ყველა მახასიათებელი ინახება როგორც წყვილი <სახელი, ლიტიპიზირებული მნიშვნელობა>. ეს ნიშნავს, რომ ერთი ცხრილის ობიექტის მახასიათებებლები შეიძლება საგრძნობლად განსხვავდებოდნენ. ცხრილში შეიძლება იყოს ორი ობიექტიც, რომელთა მახასიათებლებს ერთნაირი სახელი, მაგრამ სხვადასხვა მნიშვნელობათა ტიპები ჰქონდეთ.
- ყველა ობიექტის მონაცემთა საერთო მოცულობა არ უნდა აღემატებოდეს 1 მბ. აქ შედის მახასიათებლების სახელების ზომა, ასევე მახასითებლებლების ან მათი ტიპების მნიშვნელობათა ზომა, ორი აუცილებელი გასაღების მახასიათებელის ჩათვლით (PartitionKey და RowKey)
- მხარდაჭერილი სახეობებია Binary, Bool, DateTime, Double, GUID, Int, Int64, String. შეზღუდვები ნაჩვენებია ცხრილში.

ცხრილი 3

მახასიათებელის ტიპი	აღწერა
Binary	ბაიტების მასივი 64 კბ-მდე.
Bool	ბულევო მნიშვნელობები.
DateTime	64-ბიტიანი მნიშვნელობა, რომელიც დროის UTC ფორმატშია. მხარდაჭერილი სპექტრის მნიშვნელობებია: 1/1/1600-დან 12/31/9999-მდე.
Double	64 -ბიტიანი მნიშვნელობა მცურავი წერტილით.
GUID	128-ბიტიანიგლობალური უნიკალური იდენტიფიკატორი.
Int	32-ბიტიანი მთელი მნიშვნელობა.
Int64	64-ბიტიანი მთელი მნიშვნელობა.
String	16 ბიტიანი UTF- კოდირებული მნიშვნელობა. სტრიქონური სიდიდის ზომა შეიძლება 64 კბ-მდე იყოს.

Windows Azure Table უზრუნველყოფს ცხრილების სკალარულობის შესაძელებლობას შენახვის ათასობით კვანძამდე ცხრილში ობიექტების განაწილების გზით. ობიექტების განაწილებისას სასურველია უზუნველყოთ, რომ ერთ სიმრავლეში შემავალი ობიექტები

განთავსდნენ შენახვის ერთ კვანძში. აპლიკაცია ქმნის ამ სიმრავლეებს ობიექტების PartitionKey მახასითებლების მნიშვნელობათა შესაბამისად.

აპლიკაციებისთვის ცნობილი უნდა იყოს ყოველი ცალკე აღებული სექციის სამუშაო დატვირთვა. სასურველი შედეგების უზრუნველსაყოფად, ტესტირებამ უნდა მოახდინოს მაქსიმალური სამუშაო დატვირთვის მოდელირება.

Partition Key **Document Name**	Row Key **Version**	Property 3 **Modification Time**	.....	Property N **Description**	
Examples Doc	V1.0	8/2/2007	.....	Committed version	**Partition 1**
Examples Doc	V2.0.1	9/28/2007		Alice's working version	
FAQ Doc	V1.0	5/2/2007		Committed version	
FAQ Doc	V1.0.1	7/6/2007		Alice's working version	**Partition 2**
FAQ Doc	V1.0.2	8/1/2007		Sally's working version	

ნახ. 29. სექციების მაგალითები.

ზემოთ, ნახატზე, წარმოდგენილია ცხრილი, რომელიც შეიცავს დოკუმენტების მრავალ ვერსიას. ამ ცხრილის თითოეული ობიექტი შეესაბამება განსაზღვრული დოკუმენტის განსაზღვრულ ვერსიას. ამ მაგალითში ცხრილის სექციის გასაღებს წარმოადგეს დოკუმენტის სახელი, ხოლო სტრიქონის გასაღებს - ვერსიის ნომერი. დოკუმენტის სახელი და ვერსია უნიკალურად ახდენენ ცხრილის თითოეული ობიექტის იდენტიფიცირებას. მოცემულ მაგალითში ერთი დოკუმენტის ყველა ვერსია ქმნის სექციას.

შენახვის სისტემის კარგი სკალარულობა მიიღწევა სექციების შენახვის კვანძების სიმრავლეებად განაწილებით ხარჯზე.

სისტემა თვალყურს ადევნებს სექციების გამოყენების ხასიათს და ავტომატურად თანაბრად ანაწილებს ამ სექციებს შენახვის ყველა კვანძს შორის. ეს საშუალებას აძლევს სისტემას და აპლიკაციას მოახდინონ თავის სკალურობა ცხრილის მიმართ მოთხოვნების რაოდენობის შესაბამისად. ე.ი. თუ რომელიმე სექციაზე სხვასთან შედარებით მეტი მოთხოვნაა, სისტემა ავტომატურად დაარიგებს მათ რამდენიმე შენახვის კვანძზე, ამდენად გაანაწილებს ტრაფიკს სერვერების რამდენიმე სიმრავლეს შორის. თუმცა, სექცია, ანუ ყველა ობიექტი, რომელსაც აქვს იგივე სექციის გასაღები, იქნება ერთი კვანძის სახით. მიუხედავად

ამისა, მონაცემების მოცულობა სექციის ჩარჩოებში არ შემოიფარგლება ერთი საცავის კვანძის მოცულობით.

ერთი სექციის ობიექტები ერთად ინახება. ეს უზრუნველყოფს სექციის მიმართ მოთხოვნის ყველაზე ეფექტურ დამუშავებას. უფრო მეტიც, ამ შემთხვევაში აპლიკაციამ შეიძლება გამოიყენოს ეფექტური კეშირების უპირატესობა და პროდუქტიულობის სხვა ოპტიმიზაციები, რომელიც უზრუნველყოფილია ამ სექციაში მონაცემთა განლაგებით.

ზემოთ მოყვანილ მაგალითში სექციას ქმნის ერთი დოკუმენტის ყველა ვერსია. ამრიგად, ამ დოკუმენტის ყველა ვერსიის ამოსაწერად, თქვენ უნდა შეასრულოთ მხოლოდ ერთ სექციაზე წვდომა. გარკვეულ წლამდე მოდიფიცირებული დოკუმენტების ყველა ვერსიის მისაღებად საჭიროა გაკეთდეს რამდენიმე განყოფილების მოთხოვნა, რაც არ იქნება ეფექტური და დასჭირდება უფრო მეტი რესურსის მოხმარება, რადგან მოთხოვნის მიხედვით უნდა შემოწმდეს ყველა სექცია, რომელიც ამავე დროს შეიძლება განთავსებული იყოს სხვადასხვა შენახვის კვანძზე.

სექციის გასაღების არჩევა მნიშვნელოვანია აპლიკაციის სკალარულობის ეფექტურობის უზრუნველყოფის თვალსაზრისით. ამასთან აუცილებელია ერთ სექციაში ობიექტების განთავსებათა შორის კომპრომისის მოძებნა, რაც უზრუნველყობს მოთხოვნათა დიდ ეფექტურობას და ცხრილის სკალარულობას, რამდენადაც, რაც უფრო მეტი სექციაა ცხრილში, მით უფრო ადვილია Windows Azure Table-სთვის სერვერების სიმრავლეთა შორის დატვირთვის განაწილება.

მოცდის დროის მიხედვით ყველაზე ხშირი და კრიტიკული მოთხოვნებისათვის PartitionKey უნდა იყოს ჩართული მოთხოვნის გამოხატვის ნაწილი. მოთხოვნა, რომლეშიც მითითებულია PartitionKey, იქნება ბევრად ეფექტური, რამდენადაც ამ შემთხვევაში განიხილება მხოლოდ ერთი სექციის ობიექტები. თუ მოთხოვნის შესრულების დროს PartitionKey არაა მითიებული, აუცილებელი ობიექტის ძიებისას განიხილება ცხრილის ყველა სექცია, რაც მნიშვნელოვნად ამცირებს ეფექტურობას.

შემდეგ წარმოდგენილია ცხრილის PartitionKey-ის არჩევის ზოგიერთი რჩევა და რეკომენდაცია:

1. პირველ რიგში გამოარკვიეთ ცხრილის მნიშვნელოვანი მახასიათებლები. ეს არის მახასითებლები, რომლებიც გამოიყენება მოთხოვნის პირობებში.

2. ამ მნიშვნელოვანი მახასითებლებიდან ამოარჩიეთ პოტენციური გასაღებები:

- დომინირებული შეკითხვიდან აირჩიეთ მახასითებლები, რომლებიც პირობებშია გამოყენებული. მნიშვნელოვანია იმის გაგება, თუ რომელი მოთხოვნა იქნება აპლიკაცისთვის დომინირებული.
- ეს იქნება გასაღებების მახასითებლების საწყისი ნაკრები.
- დაალაგეთ გასაღებების მახასითებლები მოთხოვნაში მათი მიშვნელობის მიხედვით.

3. შეამოწმეთ, უზრუნველყოფს თუ არა გასაღებების მახასითებლები ობიექტის უნიკალურ იდენტიფიკაციას? თუ არა, გასაღებების ნაკრებში ჩართეთ უნიკალური იდენთიფიკატორი.

4. თუ გასაღებების მხოლოდ ერთი მახასიათებელი გვაქვთ, გამოიყენეთ ის როგორც PartitionKey.

5. თუ გასაღებების მხოლოდ ორი მახასიათებელი გვაქვთ, პირველი გამოიყენეთ როგორც PartitionKey, ხოლო მეორე - როგორც RowKey.

6. ორზე მეტი გასაღების მახასითებელის არსებობისას შეიძლება ვცადოთ მათი განაწილება ორ ჯგუფად: პირველი ჯგუფი იქნება PartitionKey, ხოლო მეორე - RowKey. ამ მიდგომით, აპლიკაციამ უნდა იცოდეს, რომ PartitionKey, მაგალითად, შედგება "-"-ად გაყოფილი ორი გასაღებისგან.

ახლა, როცა აპლიკაციას აქვს პოტენციური გასაღებების კომპლექტი, უნდა დავრწმუნდეთ, რომ შერჩეული *სექციონირების სქემა* სკალირებადია:

1. აპლიკაციების გამოყენების ინტენსივობის სტატისტიკური მონაცემებიდან გამომდინარე, განსაზღვრეთ ხომ არ მიგვიყვანს PartitionKey-ზე ზემოთ არჩეული სელექციონირება ძალიან დატვირთულ სექციებამდე, რომელებსაც ერთი სერვერი ეფექტურად ვერ მოემსახურება? ამის შემოწმება შესაძლებელია, თუ გამოვიყენებთ *ცხრილის სექციის დატვირთვაზე ტესტირებას.* ამისათვის ტექსტურ ცხრილში იქმნება არჩეული გასღებების შესაბამისი სექცია. იგი ექვემდებარება იმ პიკურ დატვირთვას, რომელიც მიიღება სავარაუდო სასარგებლო დატვირთვისა და მოთხოვნებიდან გამომდინარე. ეს საშუალებას გვაძლევს შევამოწმოთ, შეუძლია თუ არა *ცხრილის სექციას* უზრუნველყოს აპლიკაციის აუცილებელი პროდუქტიულობა.

2. თუ *ცხრილის სექცია* გაივლის *დატვირთვაზე ტესტირებას*, გასაღები სწორადაა შერჩეული.

3. თუ *ცხრილის სექცია* ვერ გაივლის *დატვირთვაზე ტესტირებას*, იპოვეთ იმ სექციის გასაღები, რომელიც უზურუნველყოფდა ობიექტების უფრო ვიწრო ქვეგანყოფილებებს. ეს შეიძლება გაკეთდეს არჩეული სექციის გასაღების მომდევნო გასაღების მახასიათებელთან გაერთიანებით, ან სექციის გასაღებად სხვა მნიშვნელოვანი მახასითებლის არჩევით. ამ ოპერაციის მიზანი უნდა იყოს დიდი რაოდენობის სექციების შექმნა, რომ არ წარმოიშვას ერთი ძალიან დიდი ან ძალინ დატვითული სექცია.

4. სისტემა ისეა დაპროექტირებული, რომ უზრუნველყოფს აუცილებელ სკლარულობას და დიდი რაოდენობით მოთხოვნათა დამუშავებას. მაგრამ ძალიან მაღალი ინტენსივობის მოთხოვნებისას მას უწევს დატვირთის ბალანსირება, რის შედეგადაც მოთხოვნებიდან ზოგიერთი მოცდის დროის გადამეტების გამო შეიძლება შეცდომით დამთავრდეს. ასეთი სახის შეცდომების შემცირებამ ან აღმოფხვრამ შეიძლება გამოიწვოს მოთხოვნათა ინტენსივობის დაწევა. ზოგადად რომ ვთქვათ, ასეთი შეცდომები იშვიათად წარმოიქმნება.

თქვენ ასევე შეგიძლიათ გააანალიზოთ შერჩეული გასაღების გაფართოება, განსაკუთრებით იმ შემთხვევაში, თუ მათი შერჩევის დროს არ არის ზუსტი ინფორმაცია მომხმარებლის ტრაფიკის მახასიათებლების შესახებ. ამ შემთხვევაში მნიშვნელოვანია აირჩეს გასაღებები, რომლებიც ადვილად გაიზრდება უფრო თხელი სექციონირების უზრუნველსაყოფად.

ცხრილებისა და ობიექტებისთვის მხარდაჭერილია შემდეგი ძირითადი ოპერაციები:

- ცხრილების ან ობიექტების შექმნა.
- ფილტრების გამოყენებით ცხრილების ან ობიექტების ამოიღება.
- ობიექტების განახლება (მაგრამ არა ცხრილის).
- ცხრილების ან ობიექტების წაშლა.

.NET აპლიკაციის ცხრილებთან მუშაობისათვის, შეგიძლიათ უბრალოდ გამოიყენოთ ADO.NET Data Services.

ქვემოთ ცხრილში მოყვანილია API-ს შემოთავაზებული ჩამონათვალი. რამდენადაც ADO.NET Data Services გამოყენება საბოლოოდ დადის REST პაკეტების გადაცემაზე, REST აპლიკაციის გამოყენება პირდაპირაა შესაძლებელი. გარდა იმისა, რომ REST უზრუნველყოფს

საცავებთან შეღწევის საშუალებას .NET ენების საშუალებით, ის ასევე საშუალებას იძლევა განვახორციელოთ ობიექტების სერიალიზაცია / დესერიალიზაციის უფრო დახვეწილი მართვა, რომელიც გამოგვადგება სხვადასხვა ტიპის ობიექტების არსებობის ან ცხრილში 255-ზე მეტი მახასითებლის და ა.შ. სცენარებთან მუშაობისას.

მაგალითი

ქვემოთ მოყვენილ მაგალითში აღწერილია ოპერაცია "Blogs"-ის ცხრილით. ამ ცხრილში ინახება ბლოგები MicroBlogging აპლიკაციისათვის.

MicroBlogging აპლიკაციაში არის ორი ცხრილი: Channels (არხები) და Blogs (ბლოგები). არსებობს არხების სია, ბლოგები ქვეყნდება განსაზღვრულ არხებში. მომხმარებლები იწერენ არხებს და ყოველდღიურად იღებენ ამ არხების ახალ ბლოგებს.

ამ მაგალითში განვიხილავთ მხოლოდ Blogs-ის ცხრილს და მოვიყვანთ მისი მეშვეობით შემდეგი ოპერაციებს მაგალითებს:

1. ცხრილის სქემის აღწერა
2. ცხრილის შექმნა
3. ბლოგის ცხრილში ჩასმა
4. ცხრილიდან ბლოგების სიის მიღება
5. ცხრილში ბლოგის გაახლება
6. ცხრილიდან ბლოგის ამოშლა

ცხრილის სქემა აღიწერება როგორც C#-კლასი. ასეთ მოდელს იყენებეს ADO.NET Data Services. სქემა ცნობილია მხოლოდ კლიენტური აპლიკაციისათვის და აიოლებს მონაცემებთან შეღწევას. სერვერი სქემას არ იყენებს.

განვიხილოთ Blogs ცხრილში შენახული Blogs-ის მახასითებლების აღწერა. ბლოგის თითოეული მახასითებელი შეიცავს შემდეგ მონაცემებს:

1. არხის სახელი (ChannelName) -არხი სადაც განთავსებულია ბლოგი.
2. განთავსების თარიღი.
3. ტექსტი (Text) - ბლოგის სხეულის შემადგენელი.
4. რეიტინგი (Rating) - ამ ბლოგის პოპულარობა.

პირველ რიგში, ყურადღება მიაქციეთ, რომ ცხრილისთვის განისაზღვრება PartitionKey, რომელიც წარმოადგენს არხის სახელს, რომლის ნაწილსაც წარმოადგენს ბლოგი, RowKey-ის

სახით გამოყენებულია ბლოგის გამოქვეყნების თარიღი. PartitionKey და RowKey - წარმოადგენენ Blogs ცხრილის გასაღებებს, მათი განახლება ხდება DataServiceKey (სერვერის მონაცემების გასაღები) კლასის ატრიბუტის მეშვეობით. ე.ი. Blogs ცხრილი სექციონირებულია არხების სახელების მიხედვით (ChannelName). ეს საშუალებას აძლევს აპლიკაციას ეფექტურად ამოიღოს მომხმარებლის მიერ გამოწერილი არხის უახლესი ბლოგები. გასაღებების გარდა მახასითებლების სახით განახლებულია მომსახურეთათვის დამახასიათებელი ატრიბუტები. ყველა მახასითებელს გააჩნია დათვლის და მინიჭების ღია (public) მეთოდები და ინახება Windows Azure Table ცხრილში. ამგვარად ქვემოთ მოყვანილ მაგალითში:

- Text და Rating ეგზემპლიარისათვის მახასიათებლები ინახება Azure ცხრილში.
- RatingAsString არ არის წარმოდგენილი, რადგან მისთვის არ არის განსაზღვრული მნიშვნელობა მინიჭების მეთოდი.
- Id არ ინახება, რადგან შეღწევის მეთოდები არ არის საჭარო.

```
[DataServiceKey("PartitionKey", "RowKey")]
public class Blog
{
// ChannelName
public string PartitionKey { get; set; }
// PostedDate
public string RowKey { get; set; }
// მომხმარებლის მიერ განსაზღვრული მახასითებლები
public string Text { get; set; }
public int Rating { get; set; }
public string RatingAsString { get; }
protected string Id { get; set; }
 }
```

შემდგომში განვიხილავთ, როგორ შევქმნათ Blogs ცხრილი საცავის სააღრიცხვო ჩანაწერისათვის. ცხრილის შექმნა ძირითად "Tables" ცხრილში ობიექტის შექმნის ანალოგიურია. ეს ძირითადი ცხრილი განსაზღვრულია საცავის თითოეული სააღრიცხვო ჩანაწერისათვის და შენახვის სააღრიცხვო ჩანაწერისათვის გამოყენებული თითოეული ცხრილის სახელი უნდა იყოს რეგისტრირებული ძირითად ცხრილში. ძირითადი ცხრილის კლასის აღწერა მოცემულია ქვემოთ, სადაც TableName (ცხრილის სახელი) მახასიათებელი წარმოადგენს შესაქმნელი ცხრილის სახელს.

```
[DataServiceKey("TableName")]
public class TableStorageTable
```

```
{
public string TableName { get; set; }
}
```

ცხრილის ფაქტობრივი შექმნა შემდეგნაირად მიმდინარეობს:

```
// Uri სერვისის: "http://<Account>.table.core.windows.net/"
DataServiceContext context = new DataServiceContext(serviceUri);
TableStorageTable table = new TableStorageTable("Blogs");
// ახალი ობიექტის დამატებით ვქმნით ახალ ცხრილს
// ძირითად ცხრილში "Tables"
context.AddObject("Tables", table);
// გამოძახების შედეგს SaveChanges წარმოადგენს გამოხმაურება სერვერის
DataServiceResponse response = context.SaveChanges();
```

serviceUri –ეს არის სერვერის ცხრილის uri, http://<აქ მითითებული სააღრიცხვო ჩანაწერის სახელი>.table.core.windows.net/. DataServiceContext (მონაცემების სერვერის კონტექსტი) - ეს არის ADO.NET მონაცემთა სერვერის ერთ-ერთი ძირითადი კლასი, რომელიც წარმოადგენს შესრულების დროის კონტექსტს სერვერისათვის. ის უზრუნველყოფს API-ის ობიექტების მოთხოვნის ჩასმის, განახლების და მოხსნისთვის querying ან LINQ ან RESTful URI-ს გამოყენებას და ინახავს მდგომარეობას კლიენტს მხარეს.

განვიხილოთ Blog ელემენტის ჩასმა. ობიექტიდ ჩასასმელად აპლიკაციამ უნდა შეასრულოს შემდეგი:

1. შექმენას ახალი C # ობიექტი და მიანიჭოს ყველა თვისება.
2. შექმენას DataServiceContext-ის ეგზემპლიარი, რომელიც წარმოადგენს სერვერთან დაკავშირებას მონაცემთა სერვისი ADO. NET-ში თქვენი საცავის სააღრიცხვო ჩანაწერისათვის.
3. დაამატოს C # ობიექტი კონტექსტში.
4. დაუკავშირდეს DataServiceContext ობიექტის SaveChanges მეთოდით სერვერზე მოთხოვნის გაგზავნას. ეს უზრუნველყოფს, რომ HTTP მოთხოვნის ერთეულ XML ფორმატში ATOM სერვერთან გაგზავნას.

თემა 11. უსაფრთხოება ღრუბელში

ღრუბლოვან გარემოში უსაფრთხოება - ეს არის ერთ-ერთი ძირითადი ასპექტი, რომელზედაც განსაკუთრებით აპერირებენ ღრუბლოვანი ინფრასტრუქტურის კრიტიკოსები. ადამიანთა უმეტესობას, მათი მონაცემების ღრუბლოვანი გარემოს კომპიუტერებზე გადატანა, სადაც მათი გაკონტროლება არ შეუძლიათ, ემოციურად დისკომფორტული ეჩვენებათ. გარდა ამისა, ეს მიდგომა მართლაც მოითხოვს შესაბამისი სტანდარტების საკითხების განხილვას და სხვადასხვა ნორმატიულ-სამართლებრივი ასპექტების გათვალისწინებას. სინამდვილეში ღრუბლოვანი გარემო არანაკლებ უკეთესად შეიძლება იყოს დაცული, ვიდრე ტრადიციული მონაცემთა დამუშავების ცენტრი. მაგრამ ღრუბლოვანი გარემოსთან მუშაობის დროს თქვენი მიდგომა ინფორმაციული უსაფრთხოების მიმართ ასევე რადიკალურად განსხვავებული იქნება იმისაგან, რასაც ტრადიციულ გარემოში ხართ მიჩვეული.

მონაცემთა ღრუბლოვან დამუშავებაზე გადასვლის გადაწყვეტილების მიღებისას, თქვენ უნდა განიხილოთ ინფორმაციულ უსაფრთხოებასთან დაკავშირებული მთელი რიგი კრიტიკული ასპექტი, მათ შორის:

- ღრუბლოვან ინფრასტრუქტურაში ორგანიზაციულ-სამართლებრივი ასპექტები, ადგილობრივ სამართალთან შესაბამისი საკითხები და სტანდარტები პრინციპულად განსხვავებული იქნება;
- ღრუბელში Amazon არ ასრებობს ისეთი ცნება, როგორიცაა „პერიმეტრი“. შესაბამისად, უსაფრთხოების პოლიტიკა რომელიც ისეა შემუშავებული, რომ ყურადღების ცენტრი მოქცეული იყოს პერიმეტრის დაცვა, ღრუბელში Amazon არ იმუშავებს. ასეთ პოლიტიკაზე არ უნდა იყოთ ორიენტირებული ისეთ ღრუბლოვან ინფრასტრუქტურებშიც კი, რომლებიც მხარს უჭერენ პერიმეტრის დაცვის ტრადიციულ მოდელს;
- თუმცა არასოდეს გამოქვეყნებულია ექსპლოიტები, რომლებიც მიმართული იქნებოდა ღრუბლოვანი ინფრასტრუქტურის უსაფრთხოების სისტემის კომპრომეტაციაზე, მონაცემთა ღრუბლოვანი საცავების ორგანიზების დროს უნდა მივყვეთ მაღალი ხარისხის რისკის უსაფრთხოების პროფილს;
- ვირტუალიზაციის ტექნოლოგიებს, როგორიცაა Xen, შეიძლება ჰქონდეთ საკუთარი სუსტი მხარეები, რამაც შეიძლება მიგვიყვანოს შემოტევის ახალ მიმართულებებამდე.

სიძნელეები, რომლებიც დაკავშირებულია საკანონმდებლო, ორგანიზაციულ-სამართლებრივ ასპექტებთან და სტანდარტიზაციის საკითხებთან

სამწუხაროდ, კანონმდებლობა და ორგანიზაციები, რომლებიც სტანდარტების შემუშავებაში პასუხისმგებლები არიან, გარკვეულ წილად „ჩამორჩნენ ცხოვრებას“ იმ საკითხებში, რომლებიც ვირტუალიზაციის გამოყენებას ეხება. მრავალი საკანონმდებლო აქტი და სტანდარტი გულისხმობს, რომ ნებისმიერი სერვერი წარმოადგენს რაღაც ფიზიკურ არსს. მრავალ შემთხვევაში ფიზიკურ და ვირტუალურ სერვერებს შორის სხვაობას არ გააჩნია ამათუიმ კანონზე ან სტანდარტზე გავლენა, რამდენადაც მათი შემუშავების მომენტში ვირტუალური სერვერების კონცეფცია არ იყო ფართოდ გავრცელებული.

ვიდრე გადახვალთ ღრუბლოვანი გამოთვლების გამოყენებაზე, თქვენ მკაფიოდ უნდა გაათვითცნობიეროთ ყველა კანონი და სტანდარტი, რომელებთანაც „მიმაგრებულია“ თქვენი აპლიკაციები და ინფრასტრუქტურა. ალბათ, თქვენი ფიზიკური ინფრასტრუქტურის ღრუბლოვანი ექვივალენტი ამ კანონებს, სტანდარტებს და სპეციფიკას დაეყრდნობა, მაგრამ საეჭვო შემთხვევებში თქვენ უნდა გაიაროთ კონსულტაცია ექსპერტებთან და მიიღოთ ცალსახა პასუხი იმაზე, ღრუბლოვან გარემოში უზრუნველყოფილი იქნება თუ არა ამ ნორმების შესაბამისობა. იმ შემთხვევებში, როდესაც „წმინდა“ ღრუბლოვანი ინფრასტრუქტურა არ შეესაბამება თქვენს მოთხოვნებს, თქვენ თითქმის ყოველთვის შეგიძლიათ შექმნათ შერეული გარემო, რომელიც ღრუბლოვანი ტექნოლოგიების გამოყენებით სარგებლის მიღების შესაძლებლობას მოგცემთ კანონების ყველა მოთხოვნების, სტანდარტების და სპეციფიკის ერთობლივი დაცვის პირობებში.

კანონების, სტანდარტების და სპეციფიკის შესაბამისობის ასპექტებთან ერთად, ღრუბლოვანი გარემო წინ წევს კიდევ ერთ საკითხს: იურიდიული პრობლემები, რომლებიც დაკავშირებული იმასთან, თუ კონკრეტულად სად ინახება თქვენი მონაცემები.

- თქვენი მონაცემები შეიძლება წარმოდგენილი იყოს სასამართლო ინსტანციებში სასამართლო უწყების საფუძველზე, ან გამოყენებული იქნას სხვა სასამართლო მოქმედებებში, რომელშიც მონაწილეობს თქვენი პროვაიდერი (და არა თქვენ) და რომლებიც მოითხვენ სხვა პროცედურებს, რომლებშიც თქვენ უშუალოდ შეიძლება ჩათრეული იქნათ.

- სხვა ქვეყნებში (კერძოდ ევროკავშირის ქვეყნებში) კანონდებლობის მიერ წარმოადგენილია საკმაოდ მკაცრი მოთხოვნები, რომელიც მიუთითებენ თუ სად და როგორ უნდა მოხდეს კონფიდენციალური მონაცემების შენახვა.

თუ ღრუბლოვანი გარემო გაიძულებთ მთლიანად გადახედოთ თქვენს წარმოდგენებს თქვენი ინფრასტრუქტურის რომელიმე ნაწილის შესახებ, მაშინ, ალბათ, ინფრასტურქტურის ასეთი კომპონენტი იქნება უსაფრთხოება. პირველი კითხვა რომელსაც კომპანია-კლიენტების ხელმძღვანელები სვამენ ასეთია: „რამდენად მიღირს საკუთარი მონაცემების შენახვაზე კოტროლის დაკარგვა?“ ამის მიუხედავად, რომ ეს საკითხი ყველას აწუხებს, მხოლოდ დამწყებები ეცნობიან მონაცემთა ღრუბლოვან დამუშავებას და ღრუბლოვან ინფრასტუქტურაზე გადასვლის შესაძლებლობებს, სინამდვილეში მასში არსებობს სხვა, უფრო მნიშვნელოვანი საკითხები, რომლებიც უსაფრთხოებასთან არის დაკავშირებული.

- ხანდახან სასამართლო პროცესები, რომლებშიც თქვენ არ ხართ ჩართული, თქვენი უსაფრთხოებისათვის საფრთხეს წარმოადგენს.
- მრავალი კანონი და სტანდარტი, რომლებიც მართავენ თქვენს IT ინფტასტრუქტურას, შემუშავებული იყო ვირტუალიზაციის გაუთვალისწინებლად.
- თვითონ ღრუბლოვან გარემოს ქსელში პერიმეტრის დაცვის იდეა კარგავს აზრს.
- მომხმარებლების სარეგისტრაციო ინფორმაციის მართვა გამოდის საინდენტიფიკაციო ინფორმაციის (identity management) მართვის სტანდარტული პროცედურების ჩარჩოებს გარეთ.

როგორც ღრუბლოვანი გარემოს სხვა ასპექტებში, უსაფრთხოება აქაც შესაძლებელია უზრუნველყოფილი იქნას, ამასთან ძალიან მაღალ დონეზე, ვიდრე მონაცემთა დამუშავების შიგა ცენტრში. ვირტუალური ეგზემპლარების ეფემერული (ცვალებადი) ბუნება მოითხოვს, რომ თქვენ აღიჭურვოთ უსაფრთხოების უზრუნველყოფის საიმედო პროცედურებით, რომელთა გარეშეც ჰოსტის ტრადიციული გარემოებები ფონს მშვენივრად გადიან. ამგვარად, ღრუბლოვან გამოთვლებზე გადასვლამ შეიძლება მიგვიყვანოს მომატებული დაცულობის კომპიუტერული ინფრასტუქტურის შექმნამდე.

მონაცემთა უსაფრთხოება

ფიზიკური უსაფრთხოება განსაზღვრავს, თუ როგორ ხორციელდება თქვენი ინფრასტრუქტურის მხარდამჭერ სერვერებზე ფიზიკური წვდომა. ჩვეულებრივ გარემოს

შესაძლოა ჯერ კიდევ ჰქონდეს ფიზიკურ უსაფრთხოების თვალსაზრისით შეზღუდვა. ბოლოსდაბოლოს, ფიზიკური სერვერები ჯერ კიდევ არსებობენ, სადღაც დგანან და მუშაობენ. როდესაც თქვენ ირჩევთ ღრუბლოვანი სერვისების პროვაიდერებს, თქვენთვის აუცილებელია გაეცნოთ ფიზიკური უსაფრთხოების უზრუნველყოფის მათ წესებს, და გაიგოთ თქვენი მხრიდან რა ზომები უნდა მიიღოთ, რომ დაიცვათ თქვენი სისტემა ფიზიკური ხარვეზებისაგან.

მონაცემთა მართვა

ყველაზე ღრმა და პრინციპული განსხვავება მონაცემთა დამუშავების ტრადიციულ ცენტრებსა და ღრუბლოვან გარემოს შორის მდგომარეობს იმაში, რომ თქვენი მონაცემები ფიზიკურად განთავსებულია სერვერებზე, რომლებიც გეკუთვნით არა თქვენ, არამედ გარეშე კომპანიებს. იმ საწარმოებმა და ორგანიზაციებმა, რომლებიც სარგებლობენ აუტსორსინგის მომსახურებით და ანდეს თავიანთი მონაცემები მართვითი მომსახურების პროვაიდერებს, ნაწილობრივ გადალახეს უფსკრული საკუთარ ფიზიკურ IT ინფრასტურქურასა და ღრუბლოვან გარემოს შორის. ღრუბლოვანი სერვერები კი ამას უმატებენ იმ სერვერების ფიზიკური დანახვის შეუძლებლობას, რომელზედაც განლაგებულია თქვენი მონაცემები. თუმცა ეს საკითხი უფრო ემოციურ სფეროს განეკუთვნება, მასთან მაინც დაკავშირებულია ზოგიერთი რეალური ბიზნეს-ამოცანები.

ძირითად პრაქტიკულ პრობლემას ის წარმოადგენს, რომ ზოგიერთ მუდმივ ფაქტორებს, რომელთაც არავითარი კავშირი არა აქვთ თქვენს ბიზნესთან, მაინც შეუძლიათ კითხვის ნიშნის ქვეშ დააყენონ თქვენი მონაცემების უსაფრთხოება. მაგალითად, თქვენს ინფრასტუქტურაში პრობლემის შექმნა შეუძლია ქვემოთ ჩამოთვლილი მოვლენებიდან ნებისმიერს.

- ფირმა, რომელიც თქვენ ღრუბლოვან სერვერებს გთავაზობთ, ცხადდება გაკოტრებულად, ხდება მისი ქონების (მათ შორის სერვერების) კონფისკაცია, და პროვაიდერი აღარ გაწვდით მომსახურებებს.
- რომელიმე მესამე მხარე, რომელსაც არავითარი ურთიერთობა არა აქვს თქვენთან (ან, უფრო აურესი, კონკურენტია), სასამართლოში უჩივის თქვენს ღრუბლოვან პროვაიდერს და სასამართლოს გადაწყვეტილებით იღებს ყველა სერვერთან წვდომას, რომელიც ღრუბლოვან პროვაიდერს ეკუთვნის.

• თქვენი პროვაიდერის უუნარობა სათანადოდ დაიცვას თავისი ინფრასტრუქტურის კომპონენეტები - განსაკუთრებით იმათი, რომლებიც შეეხება თქვენი ფიზიკური შეღწევადობის მართვას, რასაც შეუძლია გამოიწვიოს თქვენი სისტემების კომპრომატირება.

ამ პრობლემის გადაწყვეტა შეუძლია შემდეგ ზომებს, რომლებიც ნებისმიერ შემთხვევაში უნდა მიიღოთ, და რომლებსაც უმეტეს შემთხვევაში მომხმარებელიების უმეტესობა არ აფასებს: დაშიფრეთ თქვენი ინფორმაცია და შეასრულეთ დაშორებული სარეზერვო კოპირება.

• დაშიფრეთ ყველა კონფიდენციალური მონაცემი თქვენს მონაცემთა ბაზაში და მეხსიერებაში. გაშიფრეთ ეს ინფორმაცია მხოლოდ მეხსიერებაში და მხოლოდ იმ დროს, როცა არსებობს ამ მონაცემთა გამოყენების მოთხოვნილება. გარდა ამისა, დაშიფრეთ სარეზერვო ასლები და ყველა ქსელური კომუნიკაციები.

• მოძებნეთ კიდევ ერთი ღრუბლოვანი პროვაიდერი და რეგულარულად განახორციელეთ სარეზერვო კოპირების ავტომატიზებული პროცედურები (ამ მიზნით არსებობს როგორც კომერციული გადაწყვეტა, ისე გახსნილ კოდზე დაფუძნებული გადაწყეტილება). ეს როგორც მიმდინარე მონაცემების, ისე ისტორიული ინფორმაციის აღდგენის საშუალების გარანტიას იძლევა, ამასთან აღდგენა იმ დროსაცაა შესაძლებელი, თუ თქვენი პროვაიდერი საერთოდ გაქრება დედამიწიდან.

განვიხილოთ აღნიშნული ზომების გამოყენება მოვლენების განვითარების ყველა შესაძლო სცენარისათვის.

რა ხდება ღრუბლოვანი პროვაიდერის მოქმედების შეჩერების შემთხვევაში

ამ სცენარით შესაძლებელია მოვლენების სხვდასხვა ვარიანტი განვითარდეს: გაკოტრება, მოქმედების სხვა სფეროზე გადასვლისათვის ბიზნესის შეჩერება ან ელექტროენერგიის მოწოდებაში მასშტაბური და ხანგრძლივი შეფერხების გამო დიდ ხნით შეჩერება. რაც არ უნდა ხდებოდეს თქვენ რისკავთ დაკარგოთ თქვენს საწარმოო სისტემებზე წვდომა სხვა კომპანიის მოქმედების გამო. გარდა ამისა თქვენ რისკავთ იმითაც, რომ იმ ორგანიზაციას, რომლიც მართავს თქვენს მონაცემებს შეუძლია არ დაიცვას ისინი ადრე შეთანხმებული სერვისის დონის შესაბამისად.

ამ შემთხვევაში ყველაზე მნიშვნელოვან ასპექტს წარმოადგენს - დაშორებული სარეზერვო კოპირების რეგულარული განხორციელება და სარეზერვო ასლების თქვენი საწარმოო გარემოს

ფარგლებს გარეთ შენახვა. ამ ზომამ უნდა დაგიცვათ იმ ნეგატიური შედეგებისაგან, რომლებსაც გამოიწვევს თქვენი ღრუბლოვანი პროვეიდერის მუშაობის შეჩერება. უფრო კარგია კიდევ ერთი ღრუბლოვანი პროვაიდერის მომსახურებით სარგებლობა, რომლის მეშვეობითაც თქვენ შეძლებთ შემცვლელი ინფრასტრუქტურის გაშვებას.

რა ხდება, როდესაც სასამართლო უწყება იძულებულს ხდის თქვენს ღრუბლოვან პროვაიდერს გასხნას თქვენი მონაცემები

რა თქმა უნდა თუ თქვენ გარეული ხარ სასამართლო პროცესში და უწყება უშუალოდ თქვენთანაა გადამისამართებული, თქვენ ვალდებული ხართ სასამართლოს წარუდგინოთ თქვენი მონაცემები იმის მიუხედავად, თუ როგორ ზომებს იღებთ უსაფრთხოებისათვის და სად ინახება თქვენი მონაცემები - ღრუბლოვან გარემოში თუ თქვენს შიდა საკუთარ IT-ინფრასტრუქტურაში. მაგრამ აქ ჩვენ სხვა ვარიანტს განვიხილავთ - შემთხვევას, როდესაც უწყება მიმართულია თქვენს ღრუბლოვან პროვაიდერზე, და არა თქვენზე, და თქვენ არ ხართ ჩართული სასამართლო პროცესში და მასთან კავშირი არ გაქვთ.

წმინდა ტექნიკური თვალსაზრისით, სასამართლო უწყება უნდა მოიცავდეს საკითხების ვიწრო წრეს და თქვენ არ უნდა გეხებოდეთ. მაგრამ თქვენ არ შეგიძლიათ იყოთ სრულად დარწმუნებული იმაში, რომ სასამართლო უწყებები, რომლებიც ეხება კონკურენტულ ფირმებს შორის უახლესი ტექნოლოგიების სფეროში პროირიტეტების შესახებ დავას, ჩამოყალიბებულია კორექტულად - კერძოდ ისე, რომ გამოძიებაში ჩართული დაინტერესებულ მხარეთა იყოს საკმაოდ ვიწრო წრე. გარდა ამისა, თქვენ არ შეგიძლიათ დარწმუნებული იყოთ იმაშიც კი, რომ შეძლებ გაიგოთ, რომ უწყება საერთოდ გამოუშვეს თუ არა.

ამ სცენარისაგან დაგიცავთ მხოლოდ თქვენი მონაცემების სრული დაშიფვრა. უწყებამ შეიძლება მოითხოვოს, რომ თქვენმა ღრუბლოვანმა პროვაიდერმა წრუდგინოს სასამართლოს თქვენი მონაცემები და მათზე წვდომა, მაგრამ პროვაიდერს არ ექნება თქვენი გასაღები და გაშიფრისათვის გასაღები. მათ მისაღებად სასამთლომ უნდა მოგმართოთ თქვენ და გამოგიგზავნოთ უწყება. შედეგად თქვენ გექნებათ იგივე დონის კონტროლი ღრუბლოვან გარემოში თქვენს მონაცემებზე ისევე, როგორც მონაცემთა დამუშავების საკუთარ კერძო ცენტრზე.

რა ხდება, თუ ღრუბლოვან პროვაიდერს არ შეუძლია უზრუნველყოს საკუთარი ქსელი ადეკვატური დაცვის საშუალებებით

როდესაც ირჩევთ ღრუბლოვან პროვაიდერს, თქვენ უნდა გაეცნოთ იმას, თუ როგორ უზრუნველყოფენ ფიზიკურ დაცვას და როგორაა ორგანიზებული მათთან ქსელის და ცალკეული ჰოსტების დაცვა. როგორც პარადოქსალურად არ უნდა ჟღერდეს, მაგრამ ყველაზე დაცული ღრუბლოვანი პროვაიდერი ისაა, რომელზეც თქვენ ვერასოდეს ვერ მიიღებთდაცულობის შესახებ ინფორმაციას და ვერ შეძლებთ გაიგოთ, თუ სად მდებარეობს ფიზიკურად ის სერვერი რომელზედაც მუშაობს თქვენი ვირტუალური ეგზემპლარი. ალბათ, თუ თქვენ მიახლოებითაც კი მიხვდებით ამის შესახებ, მაშინ მოტივირებული გამტეხი, რომელიც მიზანმიმართულად აგროვებს თქვენს ორგანიზაციაზე ინფორმაციას, იმავე სირთულეს წააწყდება, შეეცდება რა განსაზღვროს იმ გარემოს ფიზიკური მდებარეობა, რომელიც თქვენ ჰოსტინგს გთავაზობთ.

შენიშვნა

> კომპანია Amazon არ ახმიანებს ინფორმაციას იმის შესახებ, თუ ფიზიკურად სად მდებარეობენ მათი მონაცემთა დამუშავების ცენტრები; ისინი მხოლოდ ახდენენ დეკლარირებას, რომ თითოეული ასეთი ცენტრი მდებარეობს არაფერით გამორჩეულ შენობაში ამერიკული ტიპის პერიმეტრის დაცვით. მაშინაც კი,თუ თქვენ გაიგებთ რომ მონაცემთა ბაზის ჩემი სერვერი მდებარეობს us-east-1a შეღწევადობის ზონაში, თქვენ ვერ გაიგებთ ვერც იმას სად მდებარეობს მონაცემთა დამუშავების ცენტრი, რომელიც ახდენს ამ ზონის ფორმირებას, და ვერც იმას აშშ-ს აღმოსავლეთის სანაპიროზე მდებარე შეღწევადობის ამ სამი ზონიდან რომელი წარმოადგენს us-east-1a-ს.

Amazon აქვეყნებს თავის უსაფრთხოების უზრუნველყოფის სტანდარტებს და პროცედურებს შემდეგ მისამართზე: http://aws.amazon.com. რომელი ღრუბლოვანი პროვეიდერის მომსახურებითაც არ უნდა ისარგებლოთ, თქვენ უნდა გესმოდეთ, უსაფრთხოების რომელი სტანდარტს მიჰყვებიან, და უნდა გქონდეთ მოელოდინი, რომ ისინი თქვენს მოთხოვნებს აღემატებიან კიდეც.

თუმცა პრაქტიკაში ვერავინ ვერ მოგცემთ გარანტიას, რომ თქვენი ღრუბლოვანი პროვაიდერი მართლაც უზრუნველყოფს იმ სტანდარტებს და პროცედურებს, რომლებსაც

ისინი დეკლარირებულად უჭერენ მხარს. მაგრამ თუ თქვენ მიჰყვებით რეკომენდაციებს, რომლებიც ამ თავშია მოცემული, თქვენი ინფორმაციის კონფიდენციალობა საიმედოდ დაცული იქნება თქვენი ღრუბლოვანი პროვეიდერის ზოგიერთი თანამშრომლის არაკომპეტენტურობისგანაც კი.

დაშიფრეთ ყველაფერი!

თქვენი მონაცემები ღრუბლოვან გარემოში, რა თქმა უნდა სადღაცას ინახება; უბრალოდ თქვენ არ გაგაჩნიათ ზუსტი ინფორმაცია იმის შესახებ, თუ კერძოდ სად მდებარეობენ ისინი. მაგრამ გაგაჩნიათ გარკვეული საბაზისო ცოდნა, მათ შორის:

- თქვენი მონაცემები მდებარეობენ ჰოსტური ოპერაციული სისტემის შიგნით, რომელიც მუშაობს ვირტუალურ მანქანაზე, და თქვენ ფლობთ ამ მონაცემებთან წვდომაზე კონტროლის მექანიზმს;
- ქსელური ტრაფიკი, რომლითაც თქვენ ცვლით თქვენს ვირტუალურ ეგზემპლიარებს, უჩინარია სხვა ვირტუალური ჰოსტებისათვის;
- უმეტესობა ღრუბლოვანი სერვისებისათვის მონაცემთა შენახვა, მათზე წვდომა დაცულია. მაგრამ, მრავალი მათგანი, Amazon S3-ის ჩათვლით, გაძლევთ საშუალებას გახსნათ ამ მონაცემებთან საჯარო წვდომა.

ქსელური ტრაფიკის დაშიფვრა

არა აქვს მნიშვნელობა, რამდენად სუსტია უსაფრთხოების წესები, რომლებსაც თქვენ იყენებთ ამ მომენტისათვის, მაგრამ ყველა შემთხვევაში, ალბათ, როგორც მინიმუმი, თქვენ შიფრავთ თქვენს ქსელურ ტრაფიკს - ყოველ შემთხვევაში უმეტესობას. Amazon-ის ღრუბლოვანი გარემოს განსაკუთრებული მიმზიდველობა იმაში მდგომარეობს, რომ ვირტუალურ სერვერებს არ ძალუძთ დაიჭირონ სხვა ვირტუალური სერვერების ქსელური ტრაფიკები. მიუხედავად ამისა, მაინც გირჩევთ ნუ დაეყრდნობით ამ თავისებურებას, იმიტომ რომ სხვა პროვაიდერებმა შეიძლება ეს არ უზრუნველყონ. უფრო მეტი, Amazon-საც შეუძლია სამომავლოდ შემოიტანოს რაიმე კიდევ ახალი ფუნქცია, რომლის გამო მოცემული

შესაძლებლობა მოძველებული აღმოჩნდეს. ამიტომ რეკომინდირებულია მთელი თქვენი ქსელური ტრაფიკი დაშიფროთ და არა მხოლოდ Web-ტრაფიკი.

სარეზერვო ასლების დაშიფვრა

როდესაც ახდენთ სარეზერვო კოპირებისათვის მონაცემთა იმ ნაკრების ფორმირებას, ისინი უნდა დაშიფროთ კრიპტომდგრადი ალგორითმის გამოყენებით, მაგალითად როგორიცაა retty Good Privacy (PGP).[16] ამის შემდეგ თქვენ შეგიძლიათ შეინახოთ ისინი Amazon S3-ის მსგავს ზომიერად დაცულ ღრუბლოვან გარემოში, ანდა სრულიად დაუცველ გარემოში.

დაშიფვრა იყენებს CPU რესურსებს. ამიტომ თავიდან რეკომენდებულია მოახდინოთ თქვენი ფაილების კოპირება დაუშიფრავი სახით სარეზერვო კოპირების დროებით სერვერზე, რომლის ამოცანაც მდგომარეობს შიფრირების განხორციელებაში, ხოლო შემდეგ გადაქაჩოთ სარეზერვო ასლები თქვენს მონაცემთა შენახვის ღრუბლოვან სისტემაში. სარეზერვო კოპირების შუალედური სერვერის გამოყენება საშუალებას მოგცემთ არა მხოლოდ თავი აარიდოთ CPU რესურსების თქვენს აპლიკაციების სერვერებზე და მონაცემთა ბაზაზე გამოყენებისათვის ზედმეტ ხარჯს, არამედ საშუალებას მოგცემთ შექმნათ მაღალი ხარისხის დაცვის მქონე ერთიანი სისტემა, რომელიც ინახვს თქვენ ანგარიშის მონაცემებს ღრუბლოვან საცავებში წვდომისათვის, იმის მაგივრად, რომ გადასცეს ანგარიშის მონაცემები თითოეულ სისტემას, რომელიც მოითხოვს სარეზერვო კოპირებას.

ფაილური სისტემების დაშიფვრა

თქვენს მიერ გამოყენებული თითოეული ვირტუალური სერვერი დაამონტაჟებს მონაცემთა შესანახ ეფემერულ მოწყობილობას (მაგალითად, ისეთებს, როგორიცაა განყოფილება/mnt ეგზემპალლარებზე UNIX EC2-ში) ან მონაცემთა ბლოკურ შენახვის მოწყობილობას. EC2 გარემოში ეფემერულ მოწყობილობის დაშიფვრის არ ქონა ზომიერ რისკს წარმოადგენს იმიტომ, რომ Xen EC2-ში სისტემა ეფემერულ მოწყობილობას ეგზემპალის მუშაობის დამთავრებისას ნულებით შლის. მაგრამ მონაცემთა ბლოკურ შენახვის მოწყობილობის

[16] იხ. https://secure.wikimedia.org/wikipedia/ru/wiki/PGP, https://secure.wikimedia.org/wikipedia/en/wiki/Pretty_Good_Privacy, http://www.pgp.com/, http://pgpru.com/, http://www.pgpi.org/products/pgp/versions/freeware/.

მომენტალური სურათების განთავსება ხდება Amazon S3-ში და დაუშიფრავი რჩება, თუ მათ დაშიფვრაზე სპეციალურ ზომებს არ მიიღებთ.

შენიშვნა

დაშიფვრა ყველა მის გამოხატვაში - ძვირადღირებული ზომაა. მაგრამ მათგან ყველაზე ძვირი დაშიფვრის ფორმას წარმოადგენს ფაილურ სისტემის დონეზე დაშიფვრა. სტანდარტული რეკომენდაციაა - დაიშიფროს მთლიანი ფაილური სისტემა, მაგრამ ეს ზომა შეიძლება აღმოჩნდეს ზოგიერთი აპლიკაციისათვის ზედმეტად არაპრაქტიკული. ბოლოსდაბოლოს, თქვენთვის აუცილებელია დააბალანსოთ კონკრეტული აპლიკაციის მიმართ მწარმოებლურობის მოთხოვნები და თქვენი მოთხოვნა მონაცემთა დაცულობაზე. სამწუხაროდ, დიდია იმის ალბათობა, რომ გარკვეულ დონეზე ეს მოთხოვნები დაიწყებენ კონფლიქტს ერთმანეთთან, და თქვენ კომპრომისის მოძებნა მოგიწევთ, რომელიც ღრუბლოვანი ინფრაქსტრუქტურის გამოყენების საშუალებას მოგცემთ.

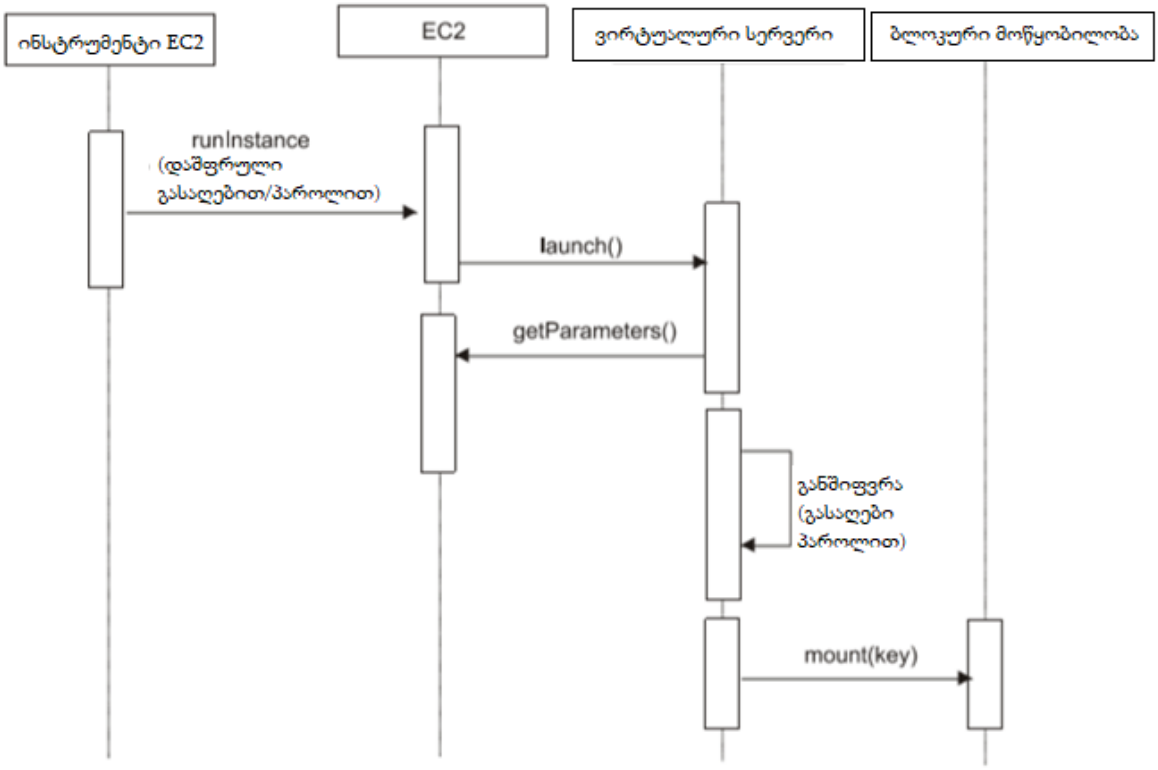
ორივე სცენართან ყველაზე უსაფრთხო მიდგომა მდგომარეობს ეფემერულ მოწყობილობის და ბლოკური შენახვის მოწყობილობების დაამონტაჟება დაშიფრული ფაილური სისტემის (encrypted filesystem) გამოყენებით. ღრუბლოვან გარემოში დაშიფრული ფაილური სისტემების გამოყენებით ვირტუალური სერვერების გაშვების მართვა უფრო მარტივია და დაცულობის შედარებით მაღალ დონეს წარმოადგენს.

სერვერებზე დაშიფრული ფაილური სისტემების გამოყენებასათან დაკავშირებული ძირითადი პრობლემა იმაში მდგომარეობს, როგორ შეძლებთ განშიფრისათვის პაროლების მართვას. კონკრეტული სერვერისათვის აუცილებელია განშიფრისათვის წინასწარ პოროლის მიღება, რითაც ის შეძლებს დაშიფრული ფაილური სისტემი დამონტაჟებას. ამ პრობლემის გადაწყვეტის ყველაზე ზოგადი მიდგომა მდგომარეობს პაროლის დაუშიფრავ ძირითად ფაილურ სისტემაში შენახვა. რამდენადაც ფაილური სისტემის მთლიანი დაშიფვრა წარმოადგენს დისკის სახესთან ფიზიკური შეღწევადობის დაცვას, პაროლის შენახვა ცალკეულ, დაუშიფრავ ფაილურ სისტემაში არაა იმდენად პრობლემატური, როგორც ეს ერთი შეხედვით ჩანს, მაგრამ მაინც პრობლემას წარმოადგენს.

ღრუბლოვან გარემოში არ გჭირდებათ ღრუბელში განშიფვრისათვის პაროლის შენახვა. ამის ნაცვლად თქვენ განშიფრისათვის პაროლი შეგიძლიათ გადასცეთ თქვენს ახალ ვირტუალურ ეგზემპლარს მისი გაშვების დროს. სერვერს შეუძლია მოიპოვოს

განშიფვრისათვის გასაღები სერვერის გაშვების ერთ-ერთი პარამეტრიდან, ხოლო შემდეგ დაამონტაჟოს ნებისმიერი ეფემერული ან ბლოკური მოწყობილობა დაშიფრული ფაილური სისტემის გამოყენებით.

თქვენ შეგიძლიათ დაამატოთ დამატებითი უსაფრთხოება პროლის დაშიფვრით და მისი განშიფვრისათვის გასაღების მანქანის გამოსახულებით შენახვით. ვირტუალური სერვერის გაშვების პროცესი, რომელიც ამონტაჟებს დაშიფრულ ფაილურ სისტემას დაშიფრული პაროლის გამოყენებით, ნაჩვენებია ნახ. 30-ზე.



ნახ. 30. ვირტუალური სერვერის გაშვების პროცესი, რომელიც ამონტაჟებს დაშიფრულ ფაილურ სისტემებს.

სტანდარტების და ნორმატიულ-საკანონმდებლო აქტებთან შესაბამისობა

უმეტესობა პრობლემებისა, რომლებიც გამოწვეულია სტანდარტების და ნორმატიულ - საკანონმდებლო აქტებთან შესაბამისობით, ძევს არა უშუალოდ ღრუბლოვან გარემოში, არამედ იმ ფაქტში, რომ ეს სტანდარტები და კონონშემოქმედი აქტები დაწერილი იყო ინტერნეტ-აპლიკაციებისათვის იქამდე სანამ ვირტუალიზაციური ტექნოლოგიები ასეთ ფართო გავრცელებას ჰპოვებდა. სხვა სიტყვებით, არსებობს ალბათობა იმისა, რომ

ღრუბლოვან ინფრასტრუქტურაში აპლიკაციების განთავსებისას, თქვენ უზრუნველყოფთ კანონების და სპეციფიკაციების შესაბამისობას, მაგრამ არა უშუალოდ მას.

მაგალითად, თუ სტანდარტი, რომლის მოთხოვნის გარანტირებაც უნდა მოახდინოთ, მოითხოვს, რომ გარკვეული მონაცემები უნდა ინახებოდეს იმ სერვერისაგან განსხვავებულ სერვერზე, რომელზედაც ხდება სისტემის ლოგიკის რეალიზება, შეუძლია კი ვირტუალურ სერვერს ასეთი შესაბამისობის უზრუნველყოფა? დარწმუნებით შეიძლება ითქვას, რომ ეს შესაძლებელია, მაგრამ თქვენი გარემოს ფაქტის ინტერპრეტაციის შესაბამისობა თუ შეუსაბამობა კანონების და სტანდარტების მოთხოვნებთან შეიძლება მიეცეს დასადგენად იურისტებს (ადვოკატებს ან მოსამართლეებს) ან სხვა ადამიანებს, რომლებიც არ არიან ტექნიკური სპეციალისტები და არ ესმით ვირტუალიზაციის ბუნება. ზოგიერთი ისეთი კანონიც კი, მაგალითად სარბაინერ-ოქსლის (Sarbanes—Oxley, SOX) [17] , რომლებიც არ წარმოადგენენ სპეციფიკურ მოთხოვნას ინფორმაციის დაცვის მიმართ, მაინც აშინებთ ტოპ-მენეჯერებს.

შეკვეცების სია:

- 95/46/EC (Directive 95/46/EC) [18] დირექტივა - ევროპის პარლამენტისა და ევროპის კავშირის საბჭოს 1997 წლის 15 დეკემბრის დირექტივა, რომელიც ეხება პერსონალური მონაცემების გამოყენებასა და პირადი ცხოვრების ხელშეუხებლობის დაცვას ტელეკომუნიკაციის სფეროში, რომელიც განსაზღვრავს „პირად მონაცემებს“ და ახდენს მათი შენახვის რეგლამენტირებას.
- სამედიცინო დაზღვევის ანგარიშგების და უსაფრთხოების კანონი (Health Insurance Portability and Accountability Act, HIPAA) [19] - ფართე კანონი, რომელიც ახედენს სამედიცინო დაზღვევასთან დაკავშირებული საკითხების რეგლამენტირებას, პირადი მონაცემების დაცვის და მათი შენახვის უსაფრთხოების ჩათვლით.

[17] სარბაიზერ-ოქსლის (Sarbanes—OxleyAct, , SOX0 ხელმოწერილი იქნა 2002 წლის 30 ივლისს და წარმოადგენს ერთ-ერთ მნიშვნელოვან მოვლენას აშშ-ს ფედერალურ კანონმდებლობაში ფასიანი ქარალდების შესახებ ბოლო 60 წლის მანძილზე. მნიშვნელოვნად ამკაცრებს ფინანსური ანგარიშგების და პროცესის მისი მომზადებისადმი მოთხოვნებს - არის შედეგი მრავალი კორპორატიული სკანდალის, რომელიც დაკავშირებული იყო მსხვილი კორპორაციების არაკეთილსინდისიერ მენეჯერებთან. დაწვრილებისთ იხ. http://tinyurl.com/35efa7r, http://tinyurl.com/3xe5qu3.

[18] დაწვრილებით იხ. https://secure.wikimedia.org/wikipedia/en/wiki/Data_Protection_Directive, http://03.rsoc.ru/docs/archive/03/documents/direktiva_97_66_ES_15.12.1997.doc.

[19] დაწვრილებით იხ. http://en.wikipedia.org/wiki/Health_Insurance_Portability_and_Accountability_Act, https://secure.wikimedia.org/wikipedia/en/wiki/Health_Insurance_Portability_and_Accountability_Act.

- გადახდების რუკის ინდუსტრიაში ინფორმაციის დაცვის სტანდარტი (Payment Card Industry Data Security Standard, PCI или PCI DSS)[20] - საერთაშორისო საგადამხდელო სისტემების Visa და MasterCard-ის მიერ შემუშავებული სტანდარტი თავისთავში მოიცავს საგადამხდელო რუკებით ტრანსაქციების დამუშავებისას ინფორმაციის დაცვისათვის რიგი პროგრამების მოთხოვნას.
- სარბანეს-ოქსლის (Sarbanes—Oxley, SOX) კანონი - ადგენს კონონმდებლურ მოთხოვნებს, რომელის დაცვაც აუცილებელია ღია სააქციო საზოგადოებების მიერ თვისი აქციონერების ინფორმირებისას.
- 21 CFR 11 (Title 21 CFR Part 11 of the Federal Code of Regulations, 21CFR11)[21] სტანდარტი — ამერიკული ორგანოს ნორმატიული დოკუმენტი საკვები პროდუქტების, სამედიცინო ტექნიკის და სამკურნალო პრეპარატების ელექტრონული ხელმოწერისა და ელექტრონული მონაცემების კონტროლის მართვის შესახებ.

უსაფრთხოების თავლსაზრისით, სტანდარტების და საკანონმდებლო აქტების მოთხოვნების უზრუნველყოფის დროს თქვენ შეიძლება წააწყდეთ შემდეგ სამ ძირითად პრობლემას.

- „როგორ“ - ამ ტიპის კითხვები შეეხება ისეთ სტანდარტებს, როგორიცაა PCI DSS ან HIPAA ან SOX -ის მსგავსი საკანონმდებლო აქტები, რომლებიც ახდენენ რეგლამენტირებას, თუ როგორ უნდა იმუშაონ კონკრეტული სფეროსთვის სპეციფიურმა განსაზღვრული ტიპის აპლიკაციებმა ინფორმაციის დაცვის მიზნით. მაგალითად, HIPAA განსაზღვრავს, თუ როგორ უნდა განხორციელდეს პერსონალური საიდენტიფიკაციო მონაცემების დამუშავება ჯანმთელობის და სამედიცინო დაზღვევის სფეროში.
- „სად“ – ეს კითხვები გამომდინარეობს ისეთი საკანონმდებლო აქტების მოთხოვნებიდან, როგორიცაა დირექტივა 95/46/EC, რომელიც განსაზღვრავს მოთხოვნებს განსაზღვრული ინფორმაციის შენახვის ადგილების მიმართ. ამ კონკრეტული დირექტივის საკვანძო გავლენის ფაქტორს წარმოადგენს ის, რომ ევროკავშირის მოქალაქების პერსონალური მონაცემები არ შეიძლება ინახებოდეს აშშ-

[20] დაწვრილებით იხ. https://secure.wikimedia.org/wikipedia/ru/wiki/PCI_DSS, http://www.osp.ru/news/articles/2010/33/13003470/

[21] დაწვრილებით იხ.https://secure.wikimedia.org/wikipedia/en/wiki/Title_21_CFR_Part_11.

ში (ან სხვა ნებისმიერ ქვეყანაში, სადაც პერსონალურ ინფორმაციის შენახვის მიმართ მოთხოვნა არ შეესაბამება მათი შენახვის ევროსაბჭოში მოქმედ მოთხოვნებს).

- „რა“ - ეს კითხვა იქმნება იმ სტანდარტების საფუძველზე, რომლებიც ითხოვენ სისტემაში იყოს სრულიად კონკრეტული კომპონენტები. მაგალითად, საგადამხდელო რუკების ინდუსტრიაში სტანდარტი მოითხოვს ანტივირუსულ პროგრამულ უზრუნველყოფას ყველა იმ სერვერზე სადაც იწარმოება საკრედიტო ბარათების მიმართ მონაცემთა დამუშავება.

დღესდღეისობით ყველაზე მნიშვნელოვან ასპექტს წარმოადგენს ის, რომ ღრუბლოვან გარემოში განთავსებული სისტემა რომელიც იცავს კანონს, შეიძლება შესაბამებოდეს სრულად ან არ შესაბამებოდეს მას. ზოგიერთი სპეციფიკაციისათვის შეიძლება უზრუნველყოფილი იყოს კანონის სრულად შესაბამისობა შერეული არქიტექტურის გამოყენების ხარჯზე, რომლის შემადგენლობაში შევლენ ზოგიერთი ფიზიკური და ზოგი ვირტუალური ელემენტები. ღრუბლოვანი ინფრასტუქტურები, რომლებიც სპეციალიზდებიან ჰიბრიდულ გადაწყვეტილებებზე, შეიძლება აღმოჩნდნენ საუკეთესო ვარიანტები. მეორეს მხრივ, შესაძლებელია აზრი ჰქონდეს, რომ ყურადღება მიექცეს იმ მომწოდებლებს, რომლებიც გთავაზობენ სერვისის სახით თქვენი სისტემის ნაწილს, რომლის გამოც თქვენ უნდა დაიცვათ ზოგიერთი სპეციფიკური მოთხოვნა. მაგალითად, თუ თქვენი საიტის ერთ-ერთი კომპონენტი ახდენს ელექტრონული კომერციის რეალიზებას, თქვენ შეგიძლიათ ისარგებლოთ ელექტრონული კომერციის სერვისების მომწოდებლებიდან იმათი მომსახურებით, რომლებიც გარანტირებულად უზრუნველყოფენ გადახდების რუკის ინდუსტრიაში ინფორმაციის დაცვის სტანდარტის მოთხოვნას (PCI DSS).

ღრუბლოვან ინფრასტუქტურაში შერეულ გარემოში თქვენ არ ინახავთ არავითარ კონფიდენციალურ მონაცემს. ამის მაგივრად თქვენ გადაგყავთ კონფიდენციალური ინფორმაციის დამუშავება ამ მიზნისათვის მონაცემთა დამუშავების ფიზიკურ ცენტრში განკუთვნილ სერვერებზე, რომლებიც მთლიანად თქვენი კონტროლის ქვეშ მდებარეობენ. მაგალითად, თქვენ შეგიძლიათ საკრედიტო ბარათების შესახებ მონაცემთა დამუშვების სერვერი გქონდეთ გარეშე მართვის მომსახურების სერვერ-პროვაიდერთან, ხოლო ამ სერვერის მიმართ მოთხოვნები (მაგალითად, საკრედიტო ბარათების ნომრების შენახვა ან ანგარიშიდან ფულის ჩამოჭრაზე მოთხოვნა) შეიძლება მოდიოდნენ ღრუბლოვანი გარემოდან.

რაც შეეხება კონფიდენციალური მონაცემების შენახვის ადგილს, Amazon გვთავაზობს S3 საცავებს, რომლებიც განთავსებულია ევროკავშირის ტერიტორიაზე. ამიტომ ღრუბლოვანი სერვისების Amazon და S3 საცავების კომბინაცია დირექტივა 95/46/EC (Directive 95/46/EC) მოთხოვნის დაცვის საშუალებას იძლევა.

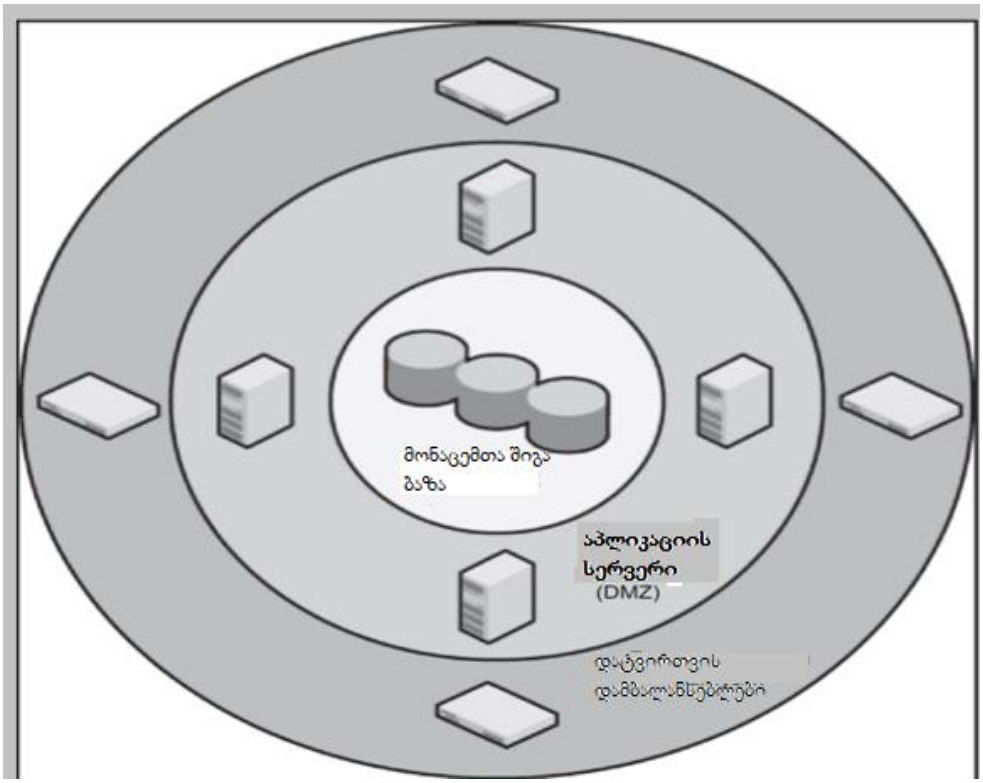
ქსელური უსაფრთხოება

როგორც ადრე ითქვა, ღრუბელ Amazon-ს პერიმეტრი არა აქვს. ამის მაგივრად EC2 წამოადგენს უსაფრთხოების ჯგუფებს, რომლებიც უზრუნველყოფენ ქსელური ტრაფიკის გაშვების წესებს, და ეს წესები ბრანდმაუერის (firewall) კონფიგურირების დროს მოცემული წესების ანალოგიურია. უსაფრთხოების ჯგუფები იმ ტრაფიკის მართვის საშუალებას იძლევიან, რომელიც ამ ჯგუფში ვირტუალურ სერვერებამდე აღწევს. თუმცა ხშირად ვახსენებთ უსაფრთხოების ჯგუფებს ისე თითქოს ისინი წარმოადგენდნენ ბრანდმაუერით დაცულ ქსელურ სეგმენტს, სინამდვილეში ისინი წარმოადგენენ ვირტუალურ ქსელურ სეგმენტებს იმიტომ, რომ:

- Amazon EC2-ის შეღწევადობის ორ სხვადასხვა ზონაში ორ სერვერს შეუძლია იმუშაოს უსაფრთხოების ერთი ჯგუფის შემადგენლობაში;
- სერვერი შეიძლება ეკუთვნოდეს უსაფრთხოების ერთზე მეტ ჯგუფს;
- უსაფრთხოების ერთ ჯგუფში შემავალ სერვერებს შეიძლება საერთოდ არ ჰქონდეთ ერთმანეთთან რაიმე კომუნიკაციის დამყარების შესაძლებლობა;
- ერთი და იმავე ქსელურ სეგმენტში მყოფ სერვერებს შეიძლება ჰქონდეთ სხვადასხვა IP მახასიათებლები - ისინი შეიძლება იმყოფებოდნენ სხვადასხვა სამისამართო სივრცეშიც;
- EC2-ში მდებარე არც ერთ სერვერს არ შეუძლია „დაინახოს“ სხვა სერვერის შემავალი და გამომავალი ქსელური ტრაფიკი (სხვა ღრუბლოვან სისტემებში არაა სავალდებულო ასე იყოს). თუ თქვენ შეეცდებით განათავსოთ თქვენი ვირტუალური სერვერი Linux აურჩეველ რეჟიმში, ე.ი. ყველა ქსელური პაკეტის მიღების რეჟიმში (promiscuous mode), მაშინ ერთადრთი ქსელური ტრაფიკი, რომლის დანახვასაც შეძლებთ, იქნება თქვენი საკუთარი შემავალი და გამომავალი ტრაფიკი.

ბრანდმაუერის წესი

როგორც წესი ბრანდმაუერი (firewall) იცავს ერთი ან რამოდენიმე ქსელური სეგმენტის პერიმეტრს. ქსელის პერიმეტრის დაცვა ბრანდმაუერის მეშვეობით ნაჩვენებია ნახ. 31-ზე



ნახ. 31. ბრანდმაუერი წარმოადგენს პერიმეტრის დაცვის ძირითად ინსტრუმენტს.

ძირითადი ბრანდმაუერი იცავს გარე პერიმეტრს, ატარებს მხოლოდ HTTP-ს, HTTPS-ს და (ხანდახან) FTP[22] -ს. დაცულ ქსელური სეგმეტსა და გარე პერიმეტრს შორის მდებარეობს სასაზღვრო სისტემები, მაგალითად დამაბალანსებელი დატვირთვები, რომლებიც მიმართავენ ტრაფიკს ეგეთ წოდებულ „დემილიტარიზებულ ზონაში“ (DMZ), რომელსაც სხვა ბრანდმაუერი იცავს. დემილიტარიზებულ ზონაში განლაგებულია ის სერვერები, რომლებიც აგზავნიან მოთხოვნებს მონაცემთა ბაზისკენ მესამე ბრანდმაუერის მეშვეობით შიგა დაცულ ქსელში, სადაც მდებარეობს შიგა მონაცემთა ბაზები, რომლებშიც კონფიდენციალური ინფორმაცია ინახება.

[22] არ ღირს საკუთარ ქსელსი ტრაფიკ FTP-ის გატარება. FTP - არ არის უსაფრთხო პროტოკოლი, და თავის ისტორიის მანძილზე სხვადასხვა რეალიზაციის დროს მრავალი ხარვეზი იქნა აღმოჩენილი. FTP-ს ნაცვლად ისარგებლეთ SCP (secure copy)- ფაილების კოპირების პროტოკოლით, რომელიც გამოიყენება ტრანსპორტის სხით არა RSH (Remote Shell), არამედ SSH (Secure Shell).

ასეთი სტრუქტურა გულისხმობს, რომ მონაცემებზე შეღწევის მისაღებად უსაფრთხოების დონის მატებასთან ერთად იქმნება ქსელური დაცვის რამდენიმე დონე (ან პერიმეტრი) ბრანდმაუერების მეშვეობით. ასეთი არქიტექტურის ძირითად უპირატესობას წარმოადგენს ის, რომ თუ ბრანდმაუერის წესი, რომელიც შიგა ქსელს იცავს ცუდადაა ფორმირებული, ისინი ვალდებული არ არიან გახსნან ის გარედან შეღწევისათვის, იმ შემთხვევების გარდა, როდესაც დემილიტარიზებული ზონაც უკვე კომპრომიტირებულია. დამატებით, საერთო ტენდენცია იმაში მდგომარეობს, რომ გარე სერვერები ინტერნეტის სუსტი მხარეებისაგან უფრო ძლიერადაა დაცული, მაშინ როდესაც შიგა სერვერები ნაკლებად ორიენტირებული არიან ინტერნეტზე. ამ ინფრასტურქტურის სისუსტე კი იმაში მდგომარეობს, რომ კონკრეტული სეგმენტის საზღვრებში ნებისმიერი შიგა სერვერიდან კომპრომეტაცია ავტომატურად წარმოადგენს სრულ შეღწევადობას ამ ქსელურ სეგმენტში სხვა სერვერებზეც.

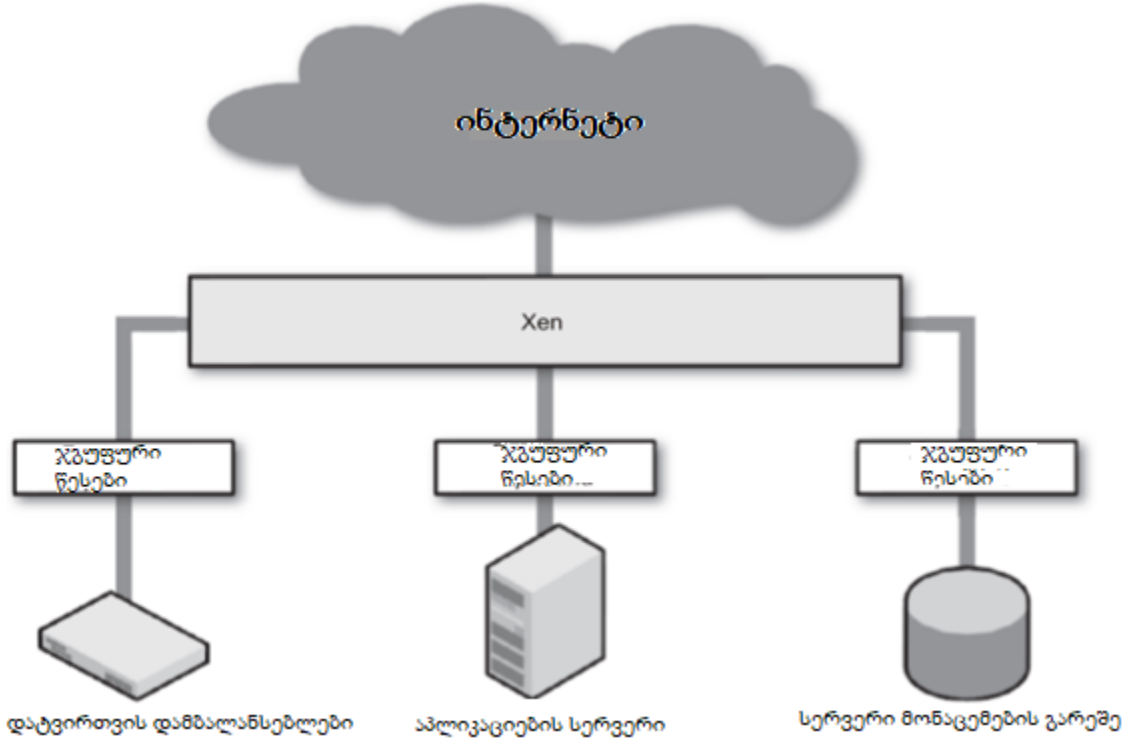
ნახ. 32-ზე წარმოდგენილია ბრანდმაუერის წესის კონცეფცია ღრუბლოვან გარემოში Amazon. როგორც ხედავთ, ღრუბლოვან ინფრასტრუქტურაში ეს კონცეფცია ძალიან განსხვავდება იმ კონცეფციისაგან, რომელიც მიღებულია მონაცემათა დამუშავების ტრადიციულ ცენტრებში.

ქსელში ყველა ვირტუალური სერვერი ერთ დონეზე მდებარეობს, ხოლო ტრაფიკით მართვა ხორციელდება *უსაფრთხოების ჯგუფის* (security group) განსაზღვრის საშუალებით. არ არსებობს არც ქსელური სეგმენტი და არც პერიმეტრი. ერთი და იმავე უსაფრთხოების ჯგუფის წევრობა არ აძლევს პრივილიგირებულ შეღწევადობას სხვას სერვერების მიმართ, რომელებიც იმავე უსაფრთხოების ჯგუფს ეკუთვნიან იმ შემთხვევის გარდა, როდესაც თქვენ ცხადად განსაზღვრავთ წესს, რომელიც გაძლევთ პრივილეგირებულ შეღწევადობას. ბოლოს, ცალკეული სერვერი შეიძლება იყოს რამდენიმე სხვადასხვა უსაფრთხოების ჯგუფის წევრი. კონტრეტული სერვერისათვის განსაზღვრული წესი წარმოადგენს ყველა იმ ჯგუფის წესების გაერთიანებას (union), რომელსაც ეს სერვერის ეკუთვნის.

უსაფრთხოების ჯგუფების განსაზღვრა იმგვარად შეგიძლიათ, რომ მოახდინოთ ქსელური პერიმერტის ტრადიციული დაცვის იმიტირება. მაგალითად, შეგიძლიათ შექმნათ შემდეგი კონფიგურაცია:

- შექმნათ უსაფრთხოების ისეთი სასაზღვრო ჯგუფი (border security group), რომელიც აწარმოებდა მთლიან ტრაფიკის მოსმენას 80 და 443 პორტების გავლით;

- შექმნათ უსაფრთხოების DMZ ჯგუფი, რომელიც გამოვა სასაზღვრო ჯგუფიდან 80 და 443 პორტების გავლით;
- შექმნათ უსაფრთხოების შიგა ჯგუფი, რომელიც აწარმოებს ტრაფიკის მოსმენას DMZ ჯგუფიდან 3306 პორტის გავლით.



ნახ. 32. ღრუბლოვან გარემოში არ არის პერიმეტრი და ქსელური სეგმენტები.

ისევე როგორც პერიმეტრის ტრადიციული დაცვის დროს შიგა უსაფრთხოების ჯგუფში არსებულ სერვერებზე შეღწევადობა, შეიძლება მიღებული იყოს მხოლოდ მაშინ, როცა თავდაპირველად ჯერ მოხდება სასაზღვო ჯგუფის კომპრომეტირება, შემდეგ - DMZ-ს, და ბოლოს - შიგა სერვერებიდან ერთ-ერთის. პერიმეტრის ტრადიციული დაცვიდან განსხვავებით, აქ არსებობს მრავალი შესაძლებლობა იმისა, რომ თქვენ შემთხვევით წარადგენთ შიგა ზონასთან გლობალურ შეღწევადობას, და ამგვარად გააღებთ მას თავდასხმისათვის. მაგრამ ღრუბლოვან გარემოში თუ თავდამსხმელი მოახდენს ერთ-ერთი შიგა ზონის სერვერის კომპრომიტირებას, არ მიიღებს ავტომატურ შეწევადობას ამ ზონის სხვა სერვერებთან თუ არ იყენებს ორიგინალურ ექსპლოიტს. სხვა სიტყვებით, შიგა ზონაში ერთ-ერთ სერვერზე წვდომა აუცილებლად არ ნიშნავს ამ ზონის ფარგლებში სხვა სერვერებზე წვდომას.

მიდგომა Amazon ისეთი შესაძლებლობების გამოყენების საშუალებას იძლევა, რომლებიც მიუღწეველი იყო ტრადიციულ ინფრასტრუქტურაში. მაგალითად, თქვენ შეგიძლიათ მარტივად უზრუნველყოთ პირდაპირი შეღწევადობა SSH-ით თქვენს ღრუბლოვან ინფრასტუქტურაში თქვენი კორპორატიული ქსელიდან ნებისმიერ ვირტუალურ სერვერთან ვირტუალური კერძო ქსელის (Virtual Private Network, VPN) გამოყენების გარეშე. ამასთან თქვენ შეინარჩუნებთ უპირატესობის გამოყენების შესაძლებლობას, რომელსაც იძლევა პერიმეტრზე ქსელის დაცვის ტრადიციული მიდგომა, როდესაც საუბარი მიდის ინტერნეტში ღია შეღწევადობაზე, და შეგიძლიათ მიიღოთ სწრაფი შეღწევადობა თქვენს სერვერთან მათი მართვის მიზნით კრიტიკული წერტილიდან.

უსაფრთხოების სისტემის ასეთი არქიტექტურა იძლევა ორ ძირითად უპირატესობას.

- რამდენადაც თქვენ შორიდან მართავთ ბრადმაურის წესებს, თავდამსხმელს არ გააჩნია ერთიანი მიზანი თავისი თავდასხმისათვის, როგორც ეს ფიზიკური ბრანდმაუერის შემთხვევაშია.
- თქვენ არ გეძლევათ საშუალება შემთხვევით დაანგრიოთ ქსელის დაცვის წესი და ამგვარად სამუდამოდ დაბლოკოთ მოცემულ ქსელურ სეგმენტში ნებისმიერი შეღწევადობა.

რეკომენდებულია მიდგომის გამოყენება, რომელიც ახდეს ქსელური პერიმეტრის ტრადიციული დაცვის იმიტაციას, იმიტომ რომ ქსელურ ტრაფიკთან ეს მიდგომა კარგადაა შესწავლილი და გასაგებად ადვილია. თუ ამ მიდგომით სარგებლობთ, მნიშვნელოვანია იმის გაგება, რომ თქვენ შექმნით მხოლოდ ტრადიციული ფიზიკური ინფრასტუქტურის ფიზიკურ ქსელურ სეგმენტის ვირტუალურ ეკვივალენტს. ქსელური უსაფრთხოების ნამდვილი დონეები, რომლებიც არსებობენ ტრადიციულ კონფიგურაციებში თქვენ არ გაქვთ.

ღრუბელში შედარებით უფრო ეფექტური უსაფრთხოების სისტემის ორგანიზების რეკომენდაციები შემდეგზე დადის.

- *თითოეულ ვირტუალურ სერვერზე გაიშვება მხოლოდ ერთი ქსელური სერვერი (პლუს ყველა სერვერი, რომელიც ადმინისტრირებისათვისაა საჭირო).* ყოველი ახალი ქსელური სერვერი, რომელიც ახლვს სისტემას, წარმოადგენს შემოტევის ვექტორს. თუ თქვენ ყურადღებას გაამახვილებთ სერვერების სიმრავლიდან ერთზე, თქვენ შექმნით შემოტევების ვექტორების სიმრავლეს, რომლებიც ამ სერვერებზე დაცულ მონაცემებთან

შეღწევის მიღებას ან ამ სერვერის ქსელის სხვა სერვერებზე შეღწევის უფლების მიღების გამოყენების პოტენციურ საშუალებას იძლევიან.

• *არ დაუშვათ იმ მონაცემებთა ღია შეღწევა, რომელთაც გააჩნიათ საიდუმლოების მაღალი დონე.* თუ თქვენი კლიენტების მონაცემთა ბაზაზე არასანქციონირებული შეღწევის მიღება მოითხოვს აპლიკაციის სერვერის და მონაცემთა ბაზის სერვერის დატვირთვის მაბალანსირებლის კომპრომეტაციას (და ამასთან თქვენ მიჰყვებით რეკომენდაციას გაუშვათ მხოლოდ ერთი სერვერი თითოეული სერვერიდან) გამტეხს დასჭირდება მოახდინოს მთელი სამი სხვადასხვა შემოტევის ვექტორის რეალიზება, ვიდრე შეძლებს ამ მონაცემების მოპოვებას.

• *გახსენით მხოლოს ის პორტები, რომელებიც აბსოლუტურად აუცილებელია სერვისის მხარდაჭერისათვის, რომელიც კონკრეტული სერვერითაა წარმოდგენილი და მეტი არაფერი.* რა თქმა უნდა, ამ სერვერებიდან თითოეული ისე უნდა იყოს გაძლიერებული, რომ მასზე მხოლოდ ერთი სერვერი მუშაობდეს - ის, რომელიც თავდაპირველად იყო განკუთვნილი მასზე სამუშაოდ. ხანდახან ისე ხდება, რომ თქვენდაუნებურად სერვერზე უშვებთ ისეთ სერვისებს, რომლებიც თავდაპირველად არ იყო განკუთვნილი ამ სერვერზე სამუშაოდ, ან თქვენ აღმოჩნდებით სიტუაციაში, როდესაც სერვერთა შემადგენლობაში აღმოაჩენთ ექპლოიტს, რომელიც არ მოითხოვს შეღწევას root სახელით (nonroot exploit), რომელიც საშუალებას იძლევა გთავდამსხმელს გაუშვას კიდევ ერთი სერვერი ექსპლოიტის მეშვეობით, რომელიც მოითხოვს შეღწევის უფლებას სწორედ root სახელით. თვენი სამიზნე სერვერის გარდა ყველა დანარჩენთან შეღწევის ბლოკირებით, თქვენ შეძლებთ აღკვეთოთ ამ ტიპის ექსპლოიტების გამოყენება.

• *შეზღუდეთ თქვენს სერვერებთან შეღწევადობა, მარტო იმ კლიენტებს მიეცით ამის საშუალება, რომლებიც ნამდვილად ამას საჭიროებენ.* რა თქმა უნდა, თქვენი დატვირთვის დამბალანსებლებმა უნდა გახსნან Web-პორტები 80 და 443 მთელი ტრაფიკისათვის. გასხნილ შეღწევადობაში საჭიროებენ მხოლოდ ეს ორი პორტი პროტოკოლისათვის და კონკრეტული სერვერებისათვის. სხვა დანარჩენი სერვერისათვის ტრაფიკი უნდა იყოს შეზღუდული კონკრეტული გამომავალი მისამართებით ან უსაფრთხოების ჯგუფებით.

• *თუკი თქვენ არ ახორციელებთ დატვირთვის ბალანსირებას, გამოიყენეთ საპირისპირო პროქსი (reverse proxy)*[23]. საპირისპირო პროქსი წარმოადგენს Web- სერვერს, მაგალითად Apache, რომელიც ახდენს კლიენტიდან სერვერის მიმართულებით ტრაფიკის მარშრუტირებას. საპირისპირო პროქსის გამოყენების ხარჯზე თქვენ შეგიძლიათ გაართულოთ თავდამსხმელისათვის თქვენს ინფრასტრუქტურაზე თავდასხმა. პირველი, Apache და IIS ბევრად უფრო უკეთ ართმევენ „საბრძოლო დავალებას“ თავს ქსელური თავდასხმების კუთხით, ვიდრე ნებისმიერი სხვა აპლკიკაციის სერვერი, რომლის გამოყენებაც შეგიძლიათ. შედეგად ექსპლოიტის შემოღწევის ალბათობა მნიშვნელოვნად დაიწევს, ხოლო მისი გაუვნებელყოფის და „პატჩის“ (patch) გამოშვების სიჩქარე მნიშვნელოვნად იზრდება, მეორე, პროქსი-სერვერზე ექსპლოიტის გამოყენება თავდამსხმელს „ხელცარიელს“ დატოვებს - არავითარ შეღწევადობას ის არ მიიღებს. თავდამსხმელს ნებისმიერ შემთხვევაში მოუწევს დამატებითი სუსუსტეების ძებნა უშუალოდ თქვენს აპლიკაციების სერვერზე.

• *გამოიყენეთ ღრუბლოვანი გარემოს დინამიური ბუნება ქსელური უსაფრთხოებიდან პრობლემის მოხსნის ავტომატიზაციისათვის.* თქვენ მაინც გჭირდებათ გახსნათ თქვენს ბრანდმაუერზე პორტები იმისათვის, რომ გადაწყვიტოთ ზოგიერთი სერიოზული ბიზნეს-ამოცანა. შესაძლოა, თქვენ გახსენით FTP პორტი Web-სერვერზე იმიტომ, რომ თქვენს ძირითად შემკვეთს გაუჩნდა FTP-ის გავლით ანონიმური შეღწევადობის გამოყენების დაჟინებული აუცილებლობა ფაილების პაკეტური გადმოქაჩვისთვის. იმის მაგივრად, რომ ეს პორტი მუდმივად გახსნილი გქონდეთ, თქვენ შეგიძლიათ გახსნათ ის დღეღამის წინასწარ განსაზღვრულ დროს და გქონდეთ გახსნილი კლიენტთან შეთანხმებული დროის მონაკვეთში, ხოლო შემდეგ ისევ დახუროთ. შეიძლება გადაწყვიტოთ დროითი სერვერის შექმნაც, რომელიც პაკეტური ჩატვირთვისათვის FTP-სერვერის როლის შემსრულებელი იქნება, დაამუშავებს გადმოქაჩულ ფაილს და შემდეგ გააჩერებს ამ სერვერს.

[23] საპირისპირო პროქსი - ეს არის პროქსი-სერვერი, რომელიც, პირდაპირისაგან განსხვავებით, ახდენს კლიენტების მოთხოვნების რეტრანსილირებას გარეშე გარემოდან ერთ ან რამდენიმე სერვერზე, რომლეიბიც ლოგიკურად განთავსებულნი არიან შიგა ქსელში. ჩვეულებრივ საპირისპირო პროქსი-სერვერები ყენდება Web-სერვერების წინ. ხშირად გამოიყენება რამდენიმე Web-სერვერს შორის ქსელური დატვირთვების დასაბალანსირებლად და მათი უსაფრთხოების ასამაღლებლად, ასრულებს რა ბრანდმაუერის (ეკრანების სიმრავლე) როლს გამოყენებით დონეზე. დაწვრილებით იხ. http://tinyurl.com/36smmyu, http://en.wikipedia.org/wiki/Reverse_proxy.

სიაში მოყვანილი რეკომენდაციები რაიმე განსაკუთრებულ ახალს არ წარმოადგენს - ეს არის უსაფრთხოების სტანდარტული ზომები. ღრუბლოვანი გარემო წარმოგვიდგენს მათი რეალიზაციის შედარებით მარტივ გზას, და აღნიშნული ზომები თამაშობენ მნიშვნელოვან როლს თქვენი ღრუბლოვანი ინფრასტრუქტურის უსაფრთხოების უზრუნველყოფაში.

ქსელური თავდასხმების აღმომჩენი სისტემები

პერიმეტრის მიხედვით ქსელის დაცვა ხშირად მოიცავს ქსელიდან თავდასხმების აღმომჩენ სისტემებს (network intrusion detection systems, NIDS), მაგალითად ისეთებს, როგორიცაა Snort[24], რომლებიც ახორციელებენ ლოკალური ტრაფიკის მონიტორინგს ყველაფერ იმის აღმოსაჩენად, რაც არასტანდარტულად ან უბრალოდ უჩვეულოდ გამოიყურება. არასტანდარტული ტრაფიკის მაგალითის სახით შეიძლება მოვიყვანოთ:

- პორტების სკანირების მცდელობები (Port scans);
- Denial-of-Service (DoS-თავდასხმები) ტიპის მიხედვით თავდასხმები;
- ექსპლოიტების გამოყენების მცდელობები, რომლებიც ახდენენ ცნობილი სუსტი ადგილების ექსპლუატაციას.

ქსელური თავდასხმების აღმოჩენა ხორციელდება ან მთელი ტრაფიკის სისტემის გავლით მარშრუტიზაციის გზით, რომელიც მის ანალიზს აკეთებს, ან კიდევ ტრაფიკის პასიური მონიტორინგით თქვენი ერთ-ერთი კომპიუტერიდან თქვენს ლოკალურ ქსელში. ღრუბლოვან ქსელში Amazon შესაძლებელია ამ ვარიანტებიდან მხოლოდ პირველი; მეორე ვარიანტის გამოყენებას აზრი არა აქვს, რამდენადაც ნებისმიერს EC2 ეგზემპლარებიდან მხოლოდ საკუთარი ტრაფიკის „დანახვა“ შეუძლია.

ქსელური თავდასხმების აღმომჩენი სისტემების დანიშნულება

ქსელური თავდასხმების აღმომჩენი სისტემების დანიშნულება ის არის, რომ გაგაფრთხილოთ შემოტევის შესაძლებლობის შესახებ მათ დაწყებამდე, და ზოგიერთ შემთხვევაში უკვე დაწყებული შემოტევის მოგერიება. მაგრამ ღრუბლოვანი გარემო Amazon-ის

[24] Snort - ეს არის თავდასხმის აღმომფხვრელი თავისუფალი ქსელური სისტემა (Intrusion Prevention System, IPS) და ღია საწყის კოდზე (Open Source) დაფუძნებული თავდასხმის აღმომჩენი ქსელური სისტემა (Intrusion Detection System, IDS), რომელსაც აქვს უნარი შეასრულოს ოაკეტების რეგისტრაცია და რეალურ დროში განახორციელოს IP- ქსელებში ტრაფიკის ანალიზი. დაწვრილებით იხ. http://www.snort.org/, http://ru.wikipedia.org/wiki/Snort

სტრუქტურის შედეგად მრავალი ამოცანა, რომელიც NIDS-ის საშუალებით ხორციელდებოდა აზრს კარგავს. მაგალითად, NIDS ჩვეულებრივ ქსელის ადმინისტრატორს აფრთხილებს პორტერების სკანირების შესახებ, რასაც იმის მაუწყებლად თვლის, რომ თავდამსხმელი თავდასხმისათვის ემზადება. მაგრამ ღრუბელ Amazon-იში, თქვენ ალბათ უბრალოდ ვერ შეამჩნევთ პორტების სკანირების მცდელობას იმიტომ, რომ NIDS-სთვის ცნობილი იქნება მხოლოდ პროტებთან მოთხოვნების შესახებ, რომლებიც ნებადართულია თქვენი უსაფრთხოების ჯგუფების წესებით. დანარჩენი ტრაფიკი NIDS-სთვის უცნობი იქნება, და შესაბამისად, არ აღიქმება როგორც პორტების სკანირება.

ისევე როგორც პორტების სკანირებისას, ქსელური თავდასხმების აღმომჩენი სისტემები Amazon აქტიურად ავლენს „უარი მომსახურებაში“ (Denial-ofService, DoS) ტიპის შემოტევებს და ალბათ, აღმოაჩენს ასეთი შემოტევების განხორციელების მცდელობას ბევრად უფრო ადრე, ვიდრე თქვენი საკუთარი თავდასხმების აღმომჩენი პროგრამული სისტემა.

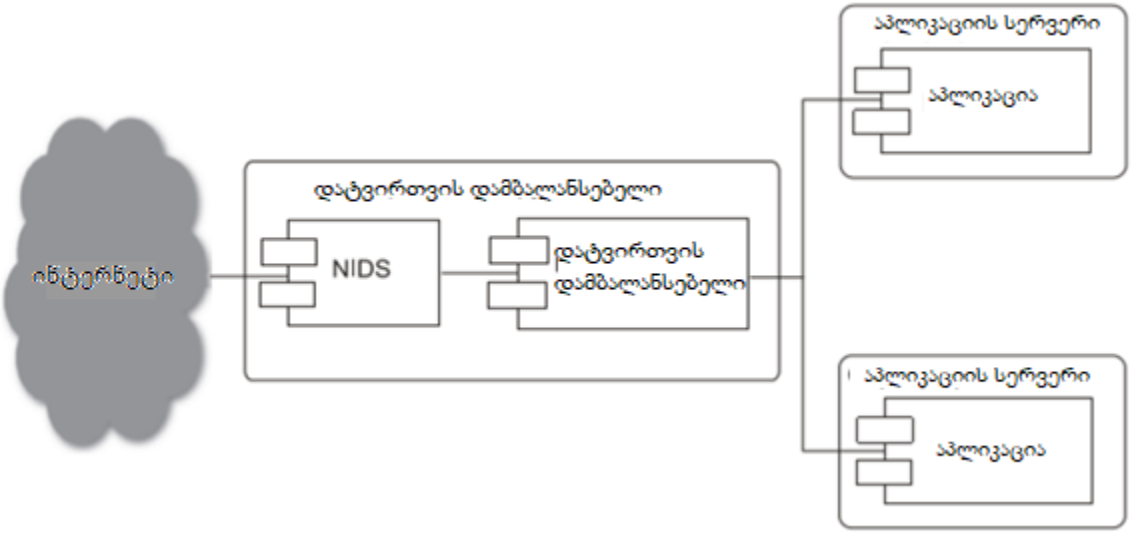
კიდევ ერთი მხარეს რომლის მიმართაც შემოტევების აღმომჩენი დამატებითი სისტემა შეიძლება აღმოჩნდეს სასარგებლო ღრუბლოვან გარემოში, წარმოადგენს მისი უნარი აღმოაჩინოს ქსელური პაკეტების დამაზიანებელი შინაარსი, რომელიც თქვენს ქსელში მოდის. როდესაც NIDS აღმოაჩენს, რომ ტრაფიკი შეიცავს დამაზიანებელ შინაარსს, მას შეუძლია დაბლოკოს ეს ტრაფიკი ან გამოგიგზავნოთ შეტყობინება, რომელიც საშუალებას მოგცემთ დროულად მოახდინოთ თავდასხმაზე რეაგირება. თუნდაც დამაზიანებელი შეტყობინება მისამართით მიღებული იქნა და შეძლებს მოახდნოს სერვერის კომპრომირება, თქვენ სწრაფად მოახდენთ რეაგირებას და თავიდან აიცილებთ თქვენზე ზიანის მიყენებას.

ღრუბლოვან გარემოში ქსელური თავდასხმების აღმომჩენი სისტემების რეალიზაცია

როგორ ზემოთ აღვნიშნეთ, ღრუბლოვან გარემოში Amazon (ისევე როგორც სხვა ღრუბლოვან სტრუქტურებში, რომელიც ლოკალური ქსელის ტრაფიკს მალავს) თქვენ უბრალოდ ვერ შეძლებთ დააყენოთ ქსელური თავდასხმების აღმომჩენი სისტემა, რომელიც პასიურად მიაყურადებს ქსელურ ტრაფიკს. ამის ნაცვლად თქვენ უნდა გაუშვათ NIDS თქვენს დატვირთვების დამბალასებლებზე ან ღრუბლოვან ინფრასტუქტურში სერვერებიდან თითოეულზე. თითოეულ ამ მიდგომას გააჩნია თავისი უპირატესობაც და ნაკლიც. მაგალითად, შეიძლება ზოგჯერ არ იყოთ ღრუბლოვან გარემოში NIDS-ის გამოყენების

მომხრე, გარდა იმ შემთვევებისა, ეს ცხადად განსაზღვრულია სტანდარტით ან კონომდებლობით, რომლის დაცვაც აუცილებელია.

გამოყოფილი NIDS სერვერის გამოყენებისადმი უმარტივეს მიდგომას წარმოადგენს ქსელის წინ გამოყოფილ სერვერზე მისი დაყენება, რომელიც ახორციელებს მთელი გამომავალი ტრაფიკის მონიტორინგს, და დამაზიანებელი მოქმედებების აღმოჩენის შემთხვევაში მოქმედებს შესაბამისად. ასეთი არქიტექტურის მაგალითი მოყვანილია ნახ. 33-ზე. რამდენადაც დატვითვის დამბალანსებელზე მუშაობს მხოლოდ პროგრამული სისტემა NIDS და სერვერი Apache, დატვითვის დამბალანსებელზე თავდასხმის პროფილი ძალიან დაბალი იქნება. NIDS სერვერის კომპრომეტაცია მოითხოვს პროგრამულ უზრუნველყოფას NIDS-ში ან სერვერში Apache სისუსტეების არსებობას (იმის სავარაუდოდ, რომ სისტემის სხვა კომპონენტების დაცვა შესაბამისად გაძლიერებულია, და არც ერთი რეალური სერვერი არ ახდენს იმ სხვა პროტების მიყურადებას, რომლებიც გახსნილია მთლიანობაში Web-სთვის).



ნახ.33. ქსელური თავდასხმების აღმომჩენი სისტემა, რომელიც ახორცილებს ტრაფიკის მონიტორინგს დატვითვის დამმბალანსებელზე.

ამ შემთხვევაში დატვირთვის დამბალანსებელი წარმოადგენს თქვენი ქსელური თავდასხმების აღმომჩენი სისტემის გაუმართაობის ერთადერთ წერტილს, იმიტომ რომ ზოგადად დატვირთვის დამბალანსებელი წარმოადგენს სისტემურ კომპონენტებს, რომლებიც ყველაზე გახსნილია თავდასხმისათვის. თქვენი დატვირთვის დამბალანსებლის კომპროტაციის საშუალების აღმოჩენისას თავდამსხმელს შეუძლია არა მარტო მოიპოვოს

დატვირთვის დამბალანსებელზე კონტროლი, არამედ მიიღოს შესაძლებლობა შეაჩეროს თავდასხმის აღმოჩენის პროცესი.

ალტერნატიული ვარიანტი მდგომარეობს დატვირთვის დამბალანსებლის სერვერზე თავდასხმების აღმომჩენი სისტემის რეალიზაციაში, რომელიც მოქმედებს როგორც შუალედური წერტილი დატვირთვის დამბალანსებელსა და სისტემის სხვა კომპონენტებს შორის. ზოგადად ასეთი მიდგომა უკეთესია, ვიდრე წინა, იმის გამოკლებით, რომ ის ღიას ტოვებს დატვირთვის დამბალანსებელს (იკვლევს მხოლოდ დატვირთვის დამბალანსებელზე გამავალ ტრაფიკს) და ამცირებს სისტემის ზოგად ხელმისაწვდომობას.

სერვერზე თავდასხმების აღმომჩენი სისტემის რეალიზაციისადმი კიდევ ერთი მიდგომაა მისი დაყენება ქსელის ყველა სერვერზე. ეს მიდგომა ერთობლიობაში ცოტათი მაღლა წევს სიტემაზე თავდასხმის ზოგად პროფილს, იმიტომ რომ თქვენ საბოლოოდ მიდიხართ იქამდე, რომ ყველა სერვერზე მუშაობს გავრცელებული პროგრამული უზრუნველყოფა. თქვენი NIDS-ს სუსტი მხარე იქამდე მიგვიყვანს, რომ ეს სისუტე გავრცელდება თქვენს ღრუბლოვან არქიტექტურაში თითოეულ სერვერზე. ასეთი არქიტექტურის პოზიტიურ თავისებურებად შეიძლება შევნიშნოთ - ის სერიოზულად უშლის ხელს თავდამსხმელებს „დაფარონ“ თავიანთი მავნებლური მოქმედების „კვალი“.

როგორც ზემოთ აღვნიშნეთ ყველა არაა ღრუბლოვან გარემო Amazon -ში სერვერზე თავდასხმების აღმომჩენი სისტემის გამოყენების მომხრე. ტრადიციულ ინფრასტრუქტურისაგან განსხვავებით, აქ არ არის NIDS-ის გამოყენების განსაკუთრებული მიზეზი, რომელიც უზრუნველყოფდა თქვენს სისტემა NIDS-ს „დაენახა“ თქვენს ეგზემპლიარისაკენ მომავალი მთელი ტრაფიკი. იმათგან საუკეთესო რისი გაკეთებაც შეგიძლიათ მდგომარეობს ისეთი რეალიზაციის შექმნაა, როდესაც NIDS ყენდება თქვენი ინფრასტუქტურის თითოეულ სერვერზე იმისათვის, რომ გქონდეთ საშულება „დაინახოთ“ მთელი ის ტრაფიკი, რომელსაც Amazon ატარებს კონკრეტული ეგზემპლარის კუთვნილ უსაფრთხოების ჯგუფში. თქვენ გექნებათ პრევენციული შეტყობინებისათვის ხელმისაწვდომი მინიმალური შესაძლებლობა, ხოლო ძირითადი უპირატესობა იქნება ქსელური პაკეტების მავნებლური შინაარსისაგან დაცვა. მაგრამ თუ თქვენ შიფრავთ მთელ ტრაფიკს, ეს უპირატესობაც კი მინიმუმამდე იქნება დაყვანილი. მეორეს მხრივ, NIDS-ის არსებობა

სერიოზულად აქვეითებს ამ სერვერების მწარმოებლურობას და ქმნის თქვენს ინფრასტრუქტურაში ყველა ჰოსტზე თავდასხმის ერთიან ვექტორს.

ჰოსტების დაცვა

ჰოსტის დაცვა აღწერს, თუ როგორაა კონფიგურირებული თქვენი სერვერი შემდეგი ამოცანების შესასრულებლად:

- თავდასხმის პრევენცია;
- თქვენს მთლიან სისტემაზე წარმატებულად ჩატარებული თავდასხმის გავლენის შემცირება ზოგადად;
- დაწყებულ თავდასხმაზე რეაქცია.

ყველა შემთხვევაში ყველაზე მომგებიანი მიდგომა მდგომარეობს ისეთი პროგრამული უზრუნველყოფის გამოყენებაში, რომელშიც არ არსებობს ხარვეზები უსაფრთხოების სისტემაში. მაგრამ ეს ხომ გამართლების საქმეა! რეალურ ცხოვრებაში, თავდასხმების თავიდან აცილების საუკეთესო მიდგომა არის იმის ვარაუდი, რომ თქვენი პროგრამული უზრუნველყოფა ჯერ კიდევ არაა მოწყვლადობისაგან დაცული. როგორც ზემოთ აღვნიშნეთ, თქვენს კონკრეტულ ჰოსტზე გაშვებული თითოეული სერვერი წარმოადგენს მოცემულ ჰოსტზე თავდასხმის ცალკეულ ვექტორს. რაც უფრო მეტია თავდასხმის ვექტორი, მით უფრო მაღალია იმის ალბათობა, რომ თავდამსხმელი აღმოაჩენს მათაგან ერთ-ერთს უსაფრთხოების ექსპლოიტის მეშვეობით. ამგვარად, თქვენ უნდა ეცადოთ, რომ თითოეულ ცალკეულ ჰოსტზე მუშაობდეს მხოლოდ მინიმალურად აუცილებელი პროგრამული უზრუნველყოფის ნაკრები.

იმის გათვალისწინებით, რომ თქვენი სერვისები დაუცველია, თქვენი სერვერების დაუცველობაზე თავდასხმის აცილების საუკეთესო საშუალებაა , „პატჩების“ სწრაფი დაყენება, რომლებიც ხურავენ ხარვეზებს უსაფრთხოების სისტემაში. სწორედ აქ ღრუბელოვანი გარემოს დინამიური ხასიათი რადიკალურად ცვლის უსაფრთხოების უზრუნველყოფისადმი მიდგომას. ტრადიციული მონაცემთა დამუშავების ცენტრში მთელი ინფრასტრუქტურის უსაფრთხოების განახლება გრძელი და სარისკო პროცესია. Cloud- ზე დაფუძნებულ გარემოში, ღრუბლოვან გარემოში მთელ თქვენ ღრუბლოვან ინფრასტრუქტურაში „პატჩების“დაყენება,

რომელებიც ფარავენ უსაფრთხოებაში არსებულ „ხვრელებს“ შედგება შემდეგი სამი მარტივი ნაბიჯისაგან:

1. უსაფრთხოების განახლებების დაყენება ყველა თქვენი AMI- სთვის.
2. შედეგების ტესტირება.
3. ყველა ვირტუალური სერვერის გადატვირთვა.

აქ თქვენი ინფრასტურქტურის მართვისათვის enStratus ან RightScale-ის მსგავსი ინსტრუმეტები დიდ მნიშვნელობას იძენენ. თუ თქვენ მოგეთხოვებათ შეასრულოთ ეს სამი ნაბიჯი ხელით, ღრუბლოვანი გარემო ხდება მუდმივი თავის ტკივილის წყარო. მაგრამ მრთვითი საშუალებები ახდენებ უსაფრთხოების სისტემების განახლების ავტომატიზირებას, მინიმუმამდე დაჰყავთ ხელის ოპერაციები, გაცდენის დრო და ადამიანური შეცდომების გამო გაცდენების პოტენციალური შესაძლებლობები.

სისტემის დაცვის გაძლიერება

თავდასხმის აღკვეთა იწყება თქვენი მანქანის სახის შექმნით. გამოცდილების შეძენასთან ერთად თქვენ შეძლებთ სხვადასხვა კონფიგურაციებით ექსპერემინტირებას და თქვენი სახეების მუდმივად გარდაქმნას. როგორც კი აღმოაჩენთ კონფიგურაციას, რომელიც კონკრეტული სერვერული პროფილისათვის კარგად მუშაობს, თქვენ შეძლებთ ამ სახის განთავსებამდე სისტემის დაცვის გაძლიერებას.

სერვერის დაცვის გაძლიერების პროცესი მდგომარეობს უსარგებლო სერვისების ბლოკირებაში და იმ სამომხმარებლო ანგარიშების ჩანაწერების გამორიცხვაში, რომლებსაც დიდი მნიშვნელობა არ გააჩნიათ. ამ პროცესის ეფექტურობის მეტად აწევა შეუძლიათ Bastille Linux-ის მსგავს ინსტრუმენტებს. როგორც კი დააყენებთ Bastille Linux-ს, თქვენ შეძლებთ ინტერაქტიური სკრიპტების შესრულებას, რომლებიც კითხვებს დაგისვამენ თქვენი სერვერის შესახებ. იმის შემდეგ რაც თქვენ უპასუხებთ ყველა შეკითხვას, სკრიპტი მოახდენს სერვისების და იმ ანგარიშების ჩანაწერების ბლოკირებას, რომლებიც არ წარმოადგენენ აბსოლუტურად აუცილებელს იმისათვის, რომ თქვენმა სერვერმა შეასრულოს მასზე დაკისრებული ამოცანები. კერძოდ, სკრიპტი დარწმუნდება, რომ თქვენს სისტემაში გაძლიერებული უსაფრთხოების უზრუნველყოფა აკმაყოფილებს შემდეგ კრიტერიუმებს:

- სერვერზე არ მუშაობს არავითარი სხვა ქსელური სერვერი, გარდა იმათი, რომლებიც აბსოლუტურად აუცილებელია ამ სერვერით წარმოდგენილი ფუნქციონალური შესაძლებლიბების მხარდასაჭერად.
- სისტემაში ბლოკირებულია ყველა სამომხმარებლი სააღრიცხვო ჩანაწერი, იმათ გარდა, რომლებიც საჭიროა იმ მომხმარებლების სერვერთან წვდიმისათვის, რომლებსაც ისინი საჭიროებენ და სხვა არავისთვის.
- სერვერზე მომუშავე ყველა პროგრამისათვის ყველა კონფიგურაციული ფაილი ისეა აწყობილი, რომ უზრუნველყოფილი იყოს უმაღლესი დონის დაცულობა.
- ყველა აუცილებელი სერვერი მუშაობს არაპრივილიგირებული სააღრიცხვო ჩანაწერის სახელით (მაგალითად, MySQL გაშვებული უნდა იქნას mysql მომხმარებლის სახელით და არა root).
- ყოველთვის, როცა ეს შესაძლებელია, სერვერები უნდა მუშაობდნენ დახურულ ფაილურ სისტემაში (restricted filesystem), მაგალითად chroot (chroot jail).

ვიდრე დაიწყებდეთ სამანქანო სახის კომპლექტაციას, აუცილებელია წაშალოთ ყველა ინტერაქტიური სააღრიცხვო ჩანაწერი და პაროლი, რომლებიც კონფიგურაციულ ფაილებში ინახება. თუმცა სამანქანო სახე ინახება დაშიფრულ ფორმატში, Amazon ინახავს დაშიფვრის გასაღებებს, რომლიც შეიძლება მან იძულებით გადასცეს მესამე მხარეს სასამართლოს უწყების გამო.

ანტივირუსული დაცვა

ზოგიერთი საკანონმდებლო აქტი და სტანდარტი მოითხოვს, რომ თქვენს სერვერებზე რეალიზებული იყოს ანტივირუსული დაცვის სისტემა. ეს მართლაც პრობლემური მოთხოვნაა იმიტომ, რომ ანტივირუსული დაცვის სისტემა ექსპლოიტია და თვითონ წარმოადგენს თავდასხმის ვექტორს, და ზოგიერთ ოპერაციულ სისტემაში ანტივირუსული პროგრამული უზრუნველყოფისათვის ექსპლოიტების პროცენტი შედარებით მაღალია ცნობილი ვირუსებისთვის.

გარკვეულ სიტუაციაში ანტივირუსული დაცვის სისტემები, რა თქმა უნდა აუცილებელია, მაშინ როცა ზოგიერთ შემთხვევაში შეიძლება წარმოადგენდნენ რისკის ფაქტორს. მაგალითად, თუ თქვენ უშვებთ იმას, რომ ატვირთოთ ფოტოები ან სხვა ფაილები თქვენს სერვერებზე,

რომლებიც შეიძლება თავის შემადგენლობაში ვირუსებს შეიცავდეს, მაშინ ეს ვირუსები შემდგომში შეიძლება მომხმარებლამდე მივიდეს, მაშინ თქვენ უბრალოდ ვალდებული ხართ გამოიყენოთ ანტივირუსული დაცვიდან რომელიმე იმისათვის, რომ გამორიცხოთ თქვენი საიტის ვირუსის გამავრცელებელ საშუალებად გამოყენება.

სამწუხაროდ, ყველა ანტივირუსული სისტემა ერთნაირად არაა შემუშავებული. ზოგიერთი მათგანი სხვებზე უკეთაა დაწერილი და იცავენ სხვა ანალოგიურ პროდუქტებზე უკეთესად. ბოლოს, ზოგიერთ სერვერს უბრალოდ არ გააჩნია სამუშაო პროფილი, რომლებიც ქმნიან ვირუსებს (viruses), „ჭიაყელებს“ (worms) და „ტროას ცხენებს“ (Trojans) თავდასხმის ვექტორებად. ამიტომ გამაღიზიანებელია სტანდარტები, საკანონმდებლო აქტები და სხვა მოთხოვნები, რომლებიც მოითხოვენ ანტივირუსული პროგრამული უზრუნველყოფით მთელი სისტემის მთლიან დაცვას.

ანტივირუსული დაცვის საკითხის განხილვისას პირველ რიგში უნდა განისაზღვროს, როგორია თქვენი საკუთარი მოთხოვნები. თუ თქვენ ვალდებული ხართ მოახდინოთ ანტივირუსული დაცვის სისტემის რეალიზება, მაშინ თქვენ, რა თქმა უნდა, უნდა შეასრულოთ ეს მოთხოვნა. ამასთან აუცილებლად უნდა მიექცეს ყურადღება შემდეგ ორ ასპექტს, რომელიც ანტივირუსული პროგრამული უზრუნველყოფის არჩევას ეხება.

- რამდენად ფართოა დაცვითი ზომების დაიპაზონი, რომელსაც გვთავაზობს არჩეული ანტივირუსული პაკეტი? სხვა სიყვებით რას უდრის ცნობილი ექსპლოიტების პროცენტი, რომელიც იფარება არჩეული ანტივირუსული პაკეტით[25]?
- რას უდრის საშუალო დროის ინტერვალი ვირუსის „ველურ მდგომარეობაში“ გამოშვებიდან იმ მომენტამდე, რომლიდანაც თქვენს მიერ გამოყენებული ანტივირუსული პროდუქტი იწყებს მისგან დაცვას?

იმის შემდეგ რაც თქვენ აირჩევთ ანტივირუსული პროგრამული უზრუნველყოფის მომწოდებელს და დააყენებთ პროდუქტს თქვენს სერვერებზე, თქვენ ვალდებული იქნებით შეინარჩუნოთ თქვენი ვირუსული სიგნატურების ბაზა მოწესრიგებულ მდგომარეობაში. შესაძლებელია თქვენ უფრო მეტად უსაფრთხოდ აღმოჩნდეთ თუ არ გექნებათ ანტივირუსული სისტემა საერთოდ, ვიდრე თუ გექნებათ ანტივირუსული პროგრამული

[25] დღესდღეისობით ვირუსების რაოდენობა იმდენად დიდია, რომ ანტივირუსული პროგრამული უზრუნველყოფის შემმუშავებელთა უმეტესობას არ შეუძლია უზრუნველყოს ყველა მათგანისაგან დაცვა. ეს მათ იციან და იმედოვნებენ, რომ თქვენ გესმით, რომ მათი პროდუქტი უზრუნველყოფს დაცვას მხოლოდ ცნობილი (გამოქვეყნებული) ვირუსებისაგან.

უზრუნველყოფის მოძველებული ვერსია და მათ მიერ გამოყენებული ვირუსული სიგნატურების მონაცემთა ბაზა.

ჰოსტის დონეზე თავდასხმების აღმომჩენი სისტემები

მაშინ როცა თავდასხმების აღმომჩენი ქსელური სისტემები ახორციელებენ ქსელური ტრაფიკის მონიტორინგს, ცდილობენ რა აღმოაჩინონ ანომალური ტრაფიკი და საეჭვო აქტივობები, *ჰოსტის დონეზე თავდასხმების აღმომჩენი სისტემები* (host intrusiondetection system, HIDS), მაგალითად, როგორიცაა OSSEC[26], თავლყურს ადევნებენ თქვენს სერვერს რაიმე უჩვეულოს აღმოჩენის მიზნით. სისტემები HIDS მრავალი მიმართებით მუშაობენ ანტივირუსული სისტემების ანალოგიურად მხოლოდ იმის გამოკლებით, რომ ისინი იკვლევენ სისტემას კომპრომეტაციის ყველა ნიშანს და გატყობინებენ ოპერაციული სისტემის ან სისტემური სერვისების ფაილების შეცვლის ყველა შემთხვევას.

ღრუბლოვან გარემოში დაყენებულ Linux-ის სისტემაში ძირითადად გამოიყენება OSSEC (http://www.ossec.net). OSSEC-ს გააჩნია ორი კონფიგურაციული პროფილი:

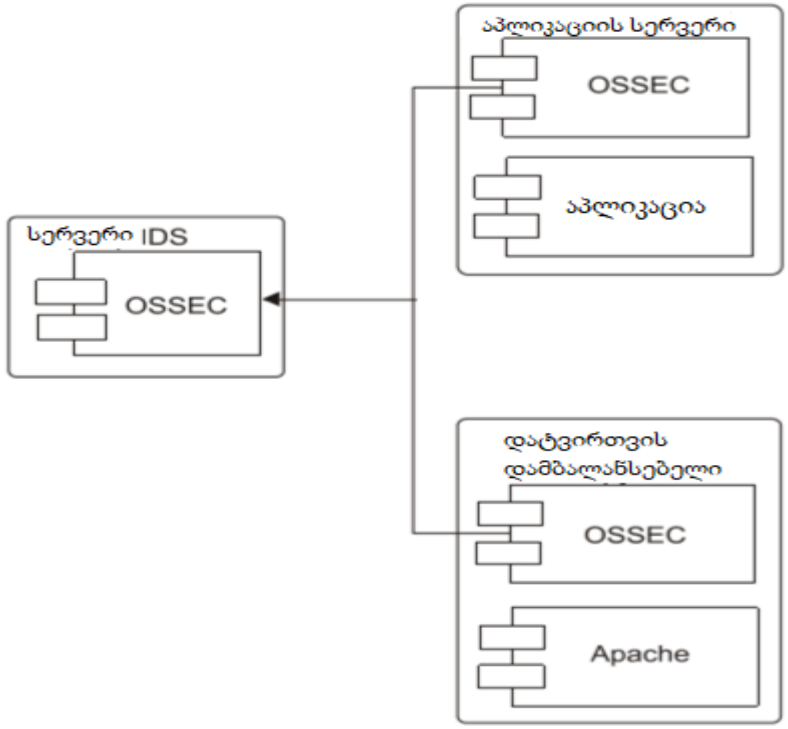
- *დამოუკიდებელი* (standalone), რომლის დროსაც თითოეული სერვერი ახდენს სკანირებას დამოუკიდებლად და გიგზავნით გაფრთხილებას;
- *ცენტრალიზებული* (centralized), რომლის დროსაც თქვენ ქმნით ცენტრალიზირებულ HIDS სერვერს, რომელზედაც დარჩენილი სარვერებიდან თითოეული აგზავნის ანგარიშს.

ღრუბლოვან ინფრასტრუქტურაში თქვენ ყოველთვის უნდა აიჩიოთ ცენტრალიზებული კონფიგურაცია. ის ახდენს ყველა თქვენი წესის ცენტრალიზებას იმისათვის, რომ სერიოზულად გაგიმარტივოთ HIDS ინფრასტრუქტურის აკურატულ მდგომარეობაში შენახვა. უფრო მეტიც, ცენტრალიზებული კონფიგურაცია საშუალებას გაძლევთ შეიმუშავოთ უსაფრთხოების მომატებული დონის პროფილი, რომელიც უზრუნველყოფს ცალკეული სერვისების დაცვის უფრო მაღალ დონეს, ვიდრე დამოუკიდებელი პროფილი. ღრუბლოვანი ინფრასტრუქტურა, რომელიც იყენებს ცენტრალიზებულ სისტემა HIDS-ს წარმოდგენილია ნახ. 35-ზე.

[26] OSSEC (Open Source Host-based Intrusion Detection System) - ეს არის ჰოსტის დონეზე თავდასხმების აღმომჩენი უფასო სისტემა, რომელიც ღია კოდზეა დაფუძნებული (Open Source).

ისევე როგორც ანტივირუსული პროგრამული უზრუნველყოფის გამოყენების შემთხვევაში, თქვენ მუდმივად უნდა ზრუნავდეთ თქვენი HIDS სერვერების მოწესრიგებულ მდგომარეობაში შენახვაზე, მაგრამ ამ დროს თქვენ არ გაქვთ ცალკეული სერვერების ისე ხშირი განახლების აუცილებლობა, როგორც ანტივირუსული პროგრამული უზრუნველყოფის გამოყენების შემთხვევაში.

HIDS-ის გამოყენებისას „მედალის მეორე მხარეს“ წარმოადგენს ის, რომ ასეთი სისტემები ძალიან მომთხოვნი არიან CPU-ის რესურსების მიმართ და, ამგვარად, შეუძლიათ მოიხმარონ თქვენი სერვერების გამოთვლითი სიმძლავრეების მნიშვნელოვანი ნაწილი. ცენტრალიზებული სერვერის მოდელის გამოყენებისას თქვენ შეგიძლიათ გადასცეთ ცენტრალურ სერვერს იმ ამოცანების დიდი ნაწილი, რომელსაც ასრულებს თავდასხმების აღმომჩენი სპეციალიზირებული სერვერი.



ნახ. 35. ცენტრალიზებულ სისტემა HIDS ღრუბლოვანი ინფრასტრუქტურაში.

მონაცემთა სეგმენტაცია

იმის დაშვებაზე დამატებით, რომ თქვენს სერვერებზე გაშვებული სერვისები მოწყვლადობისაგან არ არიან თავისუფალნი, თქვენ უნდა გამოდიოდით იქიდან, რომ ადრე თუ გვიან ერთ-ერთი მათგანი კომპრომინტირებული იქნება. რა თქმა უნდა, თქვენ არ გსურთ

ასეთი სიტუაციის წარმოქმნა. მაგრამ საუკეთესო ინფრასტრუქტურა ის იქნება, რომელიც „ტოლერანტობით“ ეპყრობა ინდივიდუალური კვანძებიდან ნებისმიერის კომპრომიტაციას, და ფაქტობრივად გულისხმობს ასეთ შესაძლებლობას. მოცემულ შემთხვევაში „ტოლერანტობის“ ქვეშ ჩვენ არ ვგულისხმობთ ცალკეული სერვისების სუსტ დაცულობას, არამედ პირიქით ისეთი მოვლენის გავლენის მინიმიზაციას, როგორიცაა ერთ-ერთი კვანძის კომპრომეტაცია, მთლიანი სისტემის ზოგად შრომისუნარიანობაზე. ასეთი მიდგომა საშუალებას მოგცემთ შეიმუშავოთ სისტემა, რომელსაც ექნება შემდეგი უპირატესობები:

- უმაღლესი დონის საიდუმლოების მქონე მონაცემებზე წვდომა მოითხოვს მთლიანი სისტემის სრულ გატეხვას;
- მთლიანი სისტემის კომპრომეტაციისათვის საჭიროა თავდასხმის მრავალი ვექტორი და კვალიფიკაციის რადიკალური სხვაობა;
- გაცდენის დრო, ასოცირებული ცალკეული კვანძის კომპრომეტაციასთან, ან უმნიშვნელოდ მცირეა ან საერთოდ ნულს უტოლდება.

მონაცემთა სეგმენტაცია კონფიდენციალობის სხვადასხვა დონესთან შესაბამისობაში სისტემაზე მთლიანობაში წარმატებულად ჩატარებული თავდასხმის გავლენის მინიმიზაციის ძირითად საშუალებას წარმოადგენს. ასეთი სისტემის რეალიზაციაში თქვენ დაგეხმარებათ მიდგომა „ერთი სერვერი-ერთი სერვისი“. რამდენადაც ამ ჯაჭვში სერვერებიდან თითოეულს ერთადერთი თავდასხმის ვექტორი გააჩნია თავდამსხმელს წარმატების მისაღწევად დასჭირდება მრავალი ექსპლოიტით სარგებლობა, რაც სერიოზულად ართულებს მის ამოცანას.

ანგარიშის ინფორმაციის მართვა

თქვენი სამანქანო სახეების პროფილები OSSEC არ უნდა შეიცავდნენ არავითარ ჩადგმულ სამომხმარებლო სააღრიცხვო ჩანაწერებს. სინამდვილეში, თქვენ არ უნდა დაუშვათ shell-თან წვდომა პაროლის გამოყენებით არც ერთ თქვენს ვირტუალურ სერვერზე. თქვენი ვირტუალური სერვერების ხელმისაწვდომობის ყველაზე უსაფრთხო მიდგომა არის ღია გასაღებების SSH დინამიური მიწოდება მიზნობრივ სერვერებზე. სხვა სიტყვებით, თუ ვინმეს სჭირდება სერვერებზე წვდომა, მაშინ თქვენ უნდა მიაწოდოთ სერვერს შესაბამისი

ინფორმაცია მისი გაშვების მომენტში ან ადმინისტრატიული ინტერფეისით, და არა სააღრიცხვო ინფორმაციის სამანქანო სახის შემადგენლობაში შენახვის ხარჯზე.

რა თქმა უნდა, SSH ღია გასაღების ჩადგმა მანქანის სახეში აბსოლუტურად უსაფრთხოა, და ეს ძალიან აადვილებს ცხოვრებას. სამწუხაროდ ეს აძნელებს ზოგადი დანიშნულების მანქანის სახის აგებას. კერთოდ, თუ თქვენ ჩააშენებთ ღია გასაღებს სააღრიცხვო ინფორმაციას მანქანის ინტერფეისში, მომხმარებელს, რომელსაც ეს ინფორმაცია შეეხება, შეუძლია მიიღოს შეღწევადობა თითოეულ ვირტუალურ სერვერთან, რომელიც ამ ინტერფეისის საფუძველზეა აგებული. იმისათვის რომ აუკრძალოთ შეღწევა ამ მომხმარებელს ან ნება დართოთ სხვას, თქვენ დაგჭირდებათ ახალი მანქანის აგება, რომელიც ასახავს თქვენთვის საჭირო ცვლილებას.

შესაბამისად თქვენ უნდა შეინარჩუნოთ იმდენად მარტივი და შესანარჩუნებლად მოსახერხებელი კონფიგურაცია, რამდენედაც ეს შესაძლებელია, სააღრიცხვო ინფორმაციის გადაცემის ხარჯზე თქვენი ვირტუალური სერვერის გაშვების პროცესში. ჩატვირთვის დროს ვირტუალურ სერვერს გააჩნია წვდომა ყველა პარამეტრთან, რომლებსაც თქვენ გადასცემთ, და მას შეუძლია შექმნას სააღრიცხვო ჩანაწერები ყველა თქვენს მიერ მითითებული მომხმარებლისათვის. ეს მარტივია, რადგან ამ ამოცანის გადაწყვეტა არ მოითხოვს დამატებით ინსტრუმენტებს იმათ გარდა, რომლებიც უკვე წარმოდგენილია Amazon-ის მიერ. მეორე მხრივ, შეღწევის ბლოკირება და მისი წარმოდგენა სისტემის ჩატვირთვის დროს ხელით გასაკეთებელ ამოცანად გადაიქცევა.

კიდევ ერთი მიდგომა ღრუბლოვანი ინფრასტრუქტურის მართვის არსებული საშუალებების გამოყენებაში ან თქვენი საკუთარი ინსტრუმენტარიების შემუშავებაში მდგომარეობს, რომელიც საშუალებას მოგცემდათ შეგენახათ მომხმარებლების სააღრიცხვო ინფორმაცია ღრუბლოვანი ინფრასტრუქტურის გარეთ და დინამიურად დაგემატებინათ ან წაგეშალათ მომხმარებლების სააღრიცხვო ჩანაწერები ღრუბლოვან სერვერებზე შესრულების დროს, მაგრამ ეს მიდგომა მოითხოვს ადმინისტრატიული სერვისის გაშვებას თითოეულ ჰოსტზე და, ამგვარად, წარმოადგენს თქვენს სერვერზე დამატებით თავდასხმის ვექტორს.

კომპრომეტაციაზე რეაქცია

რამდენადაც თქვენ უნდა გაგაჩნდეთ თავდასხმის აღმოჩენის სისტემა, თქვენ ძალიან მალე უნდა შეიტყოთ ხოლმე იმ შემთხვევების შესახებ, როდესაც ადგილი აქვს სისტემის ფაქტობრივ

კომპრომეტაციას. თუ თქვენ ასეთ სიტუაციებზე სწრაფად რეაგირებთ, თქვენ შეგიძლიათ ისარგებლოთ ექსპლოიტის გამოყენების შედეგად სისტემის კომპრომეტაციით გამოწვეული გაცდენების დროის დაწევის უპირატესობით.

როდესაც აღმოაჩენთ, რომ თქვენი ფიზიკური სერვერი კომპრომიტირებულია, რეაგირების პროცედურა წარმოადგენს გრძელ და მტკივნეულ პროცესს:

1. პირველ რიგში აუცილებელია გადაუკეტოთ შეღწევადობა თავდამსხმელს, რომელიც სისტემაში შემოიჭრა. როგორც წესი, ეს მიიღწევა სერვერის გათიშვით დანარჩენი ქსელიდან.
2. შემდეგ უნდა განისაზღვროს თავდასხმის ვექტორი. თქვენ მარტო არ მოგეთხოვებათ უბრალოდ გააჩეროთ სერვერი და გადატვირთოთ ის, რამდენედაც მოწყვლადობა, რაზეცაა საუბარი, შეიძლება შეეხოს ნებისმიერი რაოდენობის სერვერს, შემდეგ, სავსებით შესაძლებელია, რომ თავდამსხმელმა უკვე დატოვა სისტემაში რუთკიტი (rootkit)[27], ან კიდევ რაიმე პროგრამული უზრუნველყოფა, რომელიც ხელმეორედ შემოჭრის საშუალებას იძლევა იმის შემდეგ, რაც თქვენ გაანადგურებთ პირველ მოწყვლადობას, რომელმაც საშუალება მისცა შემოჭრილიყო. ამიტომ ძალიან მნიშვნელოვანია მოხდეს იმ ხერხის იდენტიფიკაცია, რომლითაც თავდამსხმელმა შეძლო თქვენი სისტემის კომპრომტაცია, ასევე გაირკვეს მისცა თუ არა ამან თქვენს ქსელში სხვა სისტემების კომპრომტაციის საშუალება, ასევე განისაზღვროს არის თუ არა სხვა სისტემებში ასეთივე მოწყვლადობა, რომელმაც თავდამსხმელს თქვენს ქსელში შემოღწევის საშუალება მისცა.
3. გამოსუფთავდეს სერვერი და შემდეგ ისევ გაიშვას, ამ საფეხურზე თქვენ დაგჭირდებათ დაადოთ „პატჩი“ (patch) ხვრელს უსაფრთხოების სისტემაში და გადააწყოთ სისტემა არაკომპრომატირებული სარეზერვო ასლის უახლოესი ვერსიის გამოყენებით.
4. გაუშვათ სერვერი მუშა მდგომარეობაში და გაიმეორთ ჩამოთვლილი საფეხურები ყველა სერვერისთვის, რომელსაც ისეთივე თავდასხმის ვექტორი გააჩნია.

ეს ძალიან შრომატევადი პროცესია და შეიძლება დიდი დრო მოითხოვოს. ღრუბლოვან გარემოში ის გაცილებით იოლად მიმდინარეობს.

პირველი, თქვენ შეგიძლიათ ჩაატაროთ ზოგიერთი პროცედურა კრიმინალისტურ ანალიზში. თვენ შეგიძლიათ ძირეული ფაილური სისტემის კოპირება ერთ-ერთი ბლოკური

[27] დაწვრილებით იხ. . http://tinyurl.com/ye4jkb6, http://en.wikipedia.org/wiki/Rootkit, http://www.z-oleg.com/secur/articles/rootkit.php, http://www.rootkit.com/.

ტომიდან, გადაიღოთ ბლოკური ტომების მომენტალური სურათი, გააჩეროთ სერვერი და მოახდინოთ მისი შეცვლა.

იმის შემდეგ, რაც შესაცვლელი სერვერი დაიწყებს მუშაობას (აუცილებლად მას ექნება იგივე მოწყვლებადობა, მაგრამ ის არ იქნება კომპრომირებული), თქვენ შეძლებთ გამოყოფილ უსაფრთხოების ჯგუფში გაუშვათ სერვერის და ჩაამონტაჟოთ კომპრომატირებული ტომები. რამდენედაც ამ ახალ სერვერს ექნება სხვა ძირეული ფაილური სისტემა, მასზე არ იმუშავებს არც ერთი სერვერი და ის არ იქნება კომპრომიტირებული. მიუხედავად ამისა თქვენ მაინც გექნებათ სრული შეღწევადობა კომპრომატირებულ ინფორმაციაზე, ასე რომ თავდასხმის ვექტორის იდენტიფიკაციის საშუალება გექნებათ. იმის შემდეგ რაც მოახდენთ თავდასხმის ვექტორის იდენტიფიკაციას, თქვენ შეძლებთ მანქანის ინტერფეისთან უსაფრთხოების განახლებას, რომელიც აღმოფხვრის მოწყვლადობას. როგორც კი მოწყვლადობა მოშორებული იქნება თქვენ გადატვირთავთ სხვა ეგზემპლარებს. ამგვარად, თქვენ უფრო ჩქარა შეძლებთ მოწყვლევადობაზე და სერვერების კომპრომეტაციაზე რეაგირებას, მინიმუმამდე (შეიძლება ნულამდეც) დაიყვანთ რა მოცდენის დრო.

გამოყენებული ლიტერატურა:

1. D.Kaleeswaran, R. Kavitha, Grid&Cloud Computing, Easwar Publishers Distributors Pvt Ltd, 2016.
2. M.Cafaro, G.Aloisio, Grids, Clouds and Virtualization, Springer, 2011.
3. K. Stanoevska-Slabeva, Th. Wozniak, S. Ristol, Grid and Cloud Computing: A Business Perspective on Technology and applications, Springer, 2010. ISBN 978-3-642-05192-0
4. D. Bakken, Smart Grids; Clouds, Communications, Open Source and Automation, CRC Press, 2014. ISBN 9781482206111.
5. L. Wang, R. Ranjan, J. Chen, B. Benatallah, Cloud Computing: Methology, Systems and Applications, CRC Press, 2011.
6. G.Reese, Cloud Aplication Architecture, O`REILLY, 2009.
7. *https://dzone.com/articles/building-integration-solutions-a-rethink*
8. *https://www.akuaroworld.com/telecom-oss-new-network-architecture/soa-model/*
9. *http://ibmwebsphereenterpriseservicebus.blogspot.com/2015/09/enterprise-requirements-for-esb.html*
10. *https://www.infoq.com/presentations/api-management-architecture*
11. *https://techglimpse.com/docker-installation-tutorial-centos/*
12. *https://martinfowler.com/articles/microservices.html*
13. *https://redmondmag.com/articles/2017/08/01/container-orchestration-with-kubernetes.aspx*
14. http://imexresearch.com/newsletters/sunblade6000c7-17.htm
15. http://www.enterprisestorageforum.com/storage-networking/storage-area-networks-in-the-enterprise.html
16. https://docs.netapp.com/ontap-9/index.jsp?topic=%2Fcom.netapp.doc.dot-cm-sanconf%2FGUID-9CD2611E-7081-4125-8418-D223C407B86E.html
17. https://www.akana.com/solutions/application_consolidation
18. http://alanquayle.com/2009/10/virtualization-technology-and/
19. https://bluelock.com/blog/virtualization-the-cloud-computing-enabler/
20. https://vittana.org/14-advantages-and-disadvantages-of-virtualization
21. https://www.hpe.com/us/en/what-is/server-virtualization.html

22. https://computer.howstuffworks.com/server-virtualization.htm

23. https://www.esds.co.in/blog/cloud-computing-types-cloud/#sthash.5mhDt8JU.ajz4STqh.dpbs

24. https://www.investopedia.com/terms/c/cloud-computing.asp

25. https://www.ripublication.com/aeee_spl/aeeev4n1spl_15.pdf