საქართველოს ტექნიკური უნივერსიტეტი

ხელნაწერის უფლებით

გიორგი ჩერქეზიშვილი

განაწილებული სისტემების პროგრამული უზრუნველყოფის ხარისხის მართვა ვირტუალური გარემოს პირობებში

დოქტორის აკადემიური ხარისხის მოსაპოვებლად
წარდგენილი დისერტაციის

ა ვ ტ ო რ ე ფ ე რ ა ტ ი

სადოქტორო პროგრამა „ინფორმატიკა“ შიფრი 0401

თბილისი

2016 წელი

სამუშაო შესრულებულია საქართველოს ტექნიკური უნივერსიტეტში
ინფორმატიკისა და მართვის სისტემების ფაკულტეტი
მართვის ავტომატიზებული სისტემების (პროგრამული ინჟინერიის)
დეპარტამენტი

ხელმძღვანელი: პროფ. გ. სურგულაძე

რეცენზენტები: პროფ. ე. თურქია
ასოც. პროფ. დ. გულუა

დაცვა შედგება ---------- წლის ”------” ----------, --------- საათზე
საქართველოს ტექნიკური უნივერსიტეტის ----------
---------- ფაკულტეტის სადისერტაციო საბჭოს კოლეგიის
სხდომაზე, კორპუსი ----------, აუდიტორია ----------
მისამართი: 0175, თბილისი, კოსტავას 77.

დისერტაციის გაცნობა შეიძლება სტუ-ს ბიბლიოთეკაში,
ხოლო ავტორეფერატისა - ფაკულტეტის ვებგვერდზე

სადისერტაციო საბჭოს მდივანი პროფ. თინათინ კაიშაური

ნაშრომის ზოგადი დახასიათება

თემის აქტუალურობა. პროგრამული უზრუნველყოფის ხარისხის მართვის პრობლემებს გასული საუკუნის 60-ანი წლებიდან მიექცა ყურადღება და დღესაც კვლავ რჩება პროგრამული ინჟინერიის ერთ-ერთ ძირითად პრობლემად. იგი განსაკუთრებული მნიშვნელობის საკითხია, რადგან საგრძნობლად გაიზარდა კომპიუტერული ტექნოლოგიების გამოყენების არეალი და გამკაცრდა მოთხოვნები მათი ფუნქციონირების საიმედოობაზე. 21-ე საუკუნის დასაწყისიდან პროგრამული უზრუნველყოფის ხარისხის მართვის პრობლემებზე აქტიურად მუშაობს აშშ-ის, ევროპის და სხვა ქვეყნების მოწინავე უნივერსიტეტები და პროგრამული სისტემების დეველოპერული ცენტრები. სახელმწიფო კორპორაციები და ბიზნესის კერძო სტრუქტურები, რომელთა ეფექტიანი ფუნქციონირება წარმოუდგენელია კომპიუტერული ტექნიკისა და ინფორმაციული ტექნოლოგიების გარეშე, ითხოვენ საიმედო პროგრამულ უზრუნველყოფას, მის უწყვეტ სრულყოფა-განვითარებას და IT-სერვისების უსაფრთხოების დონის სრულყოფას.

ბიზნეს-პროცესების ეფექტურობის, მართვადობის და გამჭვირვალობის ამაღლება, ერთიანი IT-ინფრასტრუქტურის შექმნა თანამედროვე საინფორმაციო ტექნოლოგიებით შესაძლებელია მოდელირების, დაპროექტების, პროგრამული რეალიზაციის, ტესტირებისა და ვირტუალიზაციის უახლესი მეთოდოლოგიების, მეთოდებისა და შესაბამისი ინსტრუმენტული (CASE – Computer Aided Software Engineering) საშუალებებით.

ობიექტ-ორიენტირებული მოდელირების მიდგომამ და დაპროგრამების ახალმა პარადიგმებმა საფუძველი ჩაუყარა პროგრამული სისტემების სასიცოცხლო ციკლის პროცესების სრულყოფას. ხარისხიანი გამოყენებითი პროგრამული სისტემების შექმნის და/ან გაფართოების თვალსაზრისით მნიშვნელოვანი ყურადღება ექცევა პროგრამების სასიცოცხლო ციკლის, მონაცემების და სერვერების ვირტუალიზაციის საკითხებს, რაც დღეისათვის მეტად აქტუალურია.

ასეთი პროცესების კვლევა და სრულყოფა ხელს უწყობს პროგრამული უზრუნველყოფის ხარისხის მართვის პროცესების ეფექტურობის ამაღლებას, IT-ინფრასტრუქტურის ინტენსიონალურ განვითარებას (მაგალითად, არსებული სერვერული ტექნიკის გამოყენების ეფექტიანობის ამაღლება ახალი ინვესტიციების გარეშე).

ბოლო წლების განმავლობაში ოპერაციული სისტემების ვირტუალი-ზაცია მნიშვნელოვნად განვითარდა, როგორც ტექნოლოგიურად, ასევე მარკეტინგულ საკითხშიც. ერთის მხრივ, ვირტუალიზაციის პროდუქტების მოხმარება გახდა უფრო ადვილი, უფრო საიმედო და ფუნქციონალური, ხოლო მეორეს მხრივ, აღმოჩნდა რომ ვირტუალური მანქანები შეიძლება გამოყენებულ იქნას სხვადასხვა სფეროში.

ვირტუალიზაციის გამოყენების სფერო უნდა განვსაზღვროთ როგორც „ადგილი“, სადაც არის კომპიუტერი, მაგრამ ამ დროისათვის უნდა აღვნიშნოთ ვირტუალიზაციის პროდუქტების გამოყენების შემდეგი ვარიანტები:

1. სერვერების კონსოლიდაცია:

დანართები (პროგრამული აპლიკაციები), რომლებიც მუშაობს კომპანიის

IT-ინფრასტრუქტურაში, ქმნის სერვერების აპარატულ რესურსების არცთუ ისე დიდ დატვირთვას (საშუალოდ 10-15%). ვირტუალიზაცია საშუალებას გვაძლევს მის მიგრირებას ფიზიკური სერვერებიდან ვირტუალურზე, ზრდის ჩატვირთვის მოცულობას 60-80%-ით და ამასთანავე იზრდება აპარატურის გამოყენების კოეფიციენტი. ეს გვაძლევს აპარატურის, მომსახურებისა და ელექტროენერგიის მნიშვნელოვან ეკონომიას.

ვირტუალიზაციის ბევრი პროდუქტი საშუალებას გვაძლევს ერთდროულად გავუშვათ რამდენიმე ოპერაციული სისტემა, რაც ნებას გვრთავს ჩვენ, პროგრამული უზრუნველყოფის შემქმნელებსა და ტესტერებს ჩავუტაროთ ტესტირება დანართების სხვადასხვა პლატფორმაზე და კონფიგურაციაზე. აგრეთვე, მაუსის ერთი დაკლიკვით „ფოტოების“ შექმნის მოხერებული საშუალებებისა და ამ მდგომარეობიდან აღდგენის ასეთივე მარტივი საშუალებით, ნებას გვაძლევს შევქმნათ სხვადასხვა

კონფიგურაციის სატესტო გარემო, რაც მნიშვნელოვნად ამაღლებს შექმნილი საშუალებების სიჩქარესა და ხარისხს.

ვირტუალური მანქანების გამოყენების ეს ვარიანტი არის ყველაზე ფართო მასშტაბური და შემოქმედებითი ვარიანტი. მასში ჩართულია ყველაფერი, რაც კი შეიძლება დაგვჭირდეს ბიზნესში IT რესეურსების გამოყენებისას ყოველდღიურად. მაგალითად, ვირტუალური მანქანების ბაზაზე შეგვიძლია ძალიან ადვილად შევქმნათ მუშა სადგურებისა და სერვერების სარეზერვო ასლები, ავაგოთ სისტემები, რომლებიც უზრუნველყოფს გათიშვების შემდეგ აღდგენის მინიმალურ დროს და ა.შ. ამ ჯგუფის ვარიანტებს მიეკუთვნება ყველა ის ბიზნეს-გადაწყვეტილებები, რომლებიც გამოიყენებს ვირტუალური მანქანების ძირითად უპირატესობებს.

ვირტუალიზაციის მთავარი პრინციპები:

1. საგრძნობი ეკონომიაა პროგრამული უზრუნველყოფის შეძენაზე, როდესაც ხდება რამდენიმე პროდაქშენ-სერვერის განლაგება ერთ ფიზიკურ სერვერზე. ვირტუალიზაციის პლატფორმების მომწოდებლებიდან გამომდინარე, ხელმისაწვდომია მუშა დატვირთვის ბალანსირება, მიგრაცია ფიზიკური ხოსტებისა და ბექაპს შორის. ყველაფერი ეს იწვევს სერვერების ინფრასტრუქტურის მომსახურების, მართვის და ადმინისტრირების ფულადი დანახარჯების რეალურ ეკონომიას;

2. ოპერაციული სისტემის ახალი ვერსიის გამოსვლისას, ძველი ვერსია შეიძლება შევინახოთ ვირტუალურ მანქანაზე, სანამ არ იქნება ექსპლუატაციაში გაშვებული და მთლიანად გამართული ახალი ოპერაციული სისტემა. ახალი ვერსია შეიძლება ჩავტვირთოთ ვირტუალურ მანქანაზე და გამოვცადოთ ისე, რომ რაიმე ზიანი არ მიაყენოს ძირითად სისტემას;

3. ნებისმიერ დროს მიცემული აპარატული კონფიგურაცია (პროცესორული დრო, ოპერატიული და დისკური მეხსიერების გადმოცემის

რაოდენობა) შეგვიძლია შევცვალოთ და ამ ცვლილებას გაცილებით ნაკლები დრო დასჭირდება ვიდრე ფიზიკურ სერვერზე ცვლილებას;

4. ერთ სერვერზე შეიძლება გაშვებულ იქნას რამდენიმე ვირტუალური მანქანა, რომლებიც გაერთიანებულია ვირტუალურ ქსელში. ასეთი ფუნქციები გვაძლევს უსაზღვრო შესაძლებლობებს შევქმნათ ვირტუალური ქსელის მოდელები რამდენიმე სისტემას შორის ერთ ფიზიკურ კომპიუტერზე. განსაკუთრებით ეს საჭიროა, როდესაც გვჭირდება განაწილებული სისტემis შექმნა, რომელიც შედგება რამდენიმე მანქანისაგან. აგრეთვე შეიძლება შევქმნათ რამდენიმე იზოლირებული სამომხმარებლი სფერო (სამუშაოდ, გასართობად, ინტერნეტში სამუშაოდ), გავუშვათ ის და გადავერთოთ ამა თუ იმ სფეროზე როდესაც დაგვჭირდება, რომ შევასრულოთ ესა თუ ის ამოცანა;

5. ვირტუალური მანქანები გვაძლევს ოპერაციულ სისტემებთან მუშაობის შესწავლის საშუალებას. შესაძლებელია მოხმარებისთვის გამზადებული ვირტუალური მანქანების რეპოზიტორიები სვადასხვა ოპერაციული სისტემებით და გავუშვათ ისინი საჭიროებისამებრ სწავლების მიზნით. ისინი შეიძლება გამოვიყენოთ ყველანაირად, ჩავატაროთ ექსპერიმენტები, ვინაიდან სისტემის მწყობრიდან გამოსვლის შემთხვევაში მისი აღდგენა შენახული მდგომარეობიდან შესაძლებელია რამდენიმე წუთში;

6. ვირტუალური მანქანები ზრდიან მობილურობას;

7. ფაილი შეიძლება გადავიტანოთ ვირტუალური მანქანით სხვა კომპიუტერზე. არ საჭიროებს არავითარი მიგრაციის ფორმების შექმნას და აგრეთვე ვირტუალური მანქანა არ არის დაკავშირებული რომელიმე კონკრეტულ აპარატურასთან;

8. ვირტუალური მანქანები შეიძლება თავმოყრილი იყოს სხვადასხვა „პაკეტებში“, შეიძლება შევქმნათ ვირტუალური გარემო კონკრეტული ამოცანებისთვის (მაგალითად, სატესტო გარემო, რეალური გარემო,

საბუღალტრო მანქანების გარემო და ა.შ.). მასზე ყველა აუცილებელი პროგრამული უზრუნველყოფის დაყენების შემდეგ;

9. ვირტუალური მანქანები ფიზიკურთან შედარებით უფრო მეტად მართვადია.

სამუშაოს მიზანი და ამოცანები.

დისერტაციის მიზანია განაწილებული სისტემების პროგრამული უზრუნველყოფის ხარისხის მართვა ვირტუალური გარემოს პირობებში. სერვერების კონსოლიდაცია, ბეკაპირება, რეპლიკაცია, სისტემის სათადარიგო მანქანებით უზრუნველყოფა და მათი საიმედოობა.

ვირტუალური მანქანების მართვის სისტემის საპრობლემო სფეროს საზღვრების დადგენა. მონაცემთა განაწილებული ბაზის კონცეპტუალური სქემების დაპროექტება არსთა-დამოკიდებულების მოდელით (ERM) და მისი რეალიზაცია განაწილებული ბაზის სახით Oracle პროგრამული პაკეტის საფუძველზე. თეორიულად შემუშავებული მოდელების და მეთოდების პროგრამული რეალიზაცია MsVisual_Studio.NET Framework_4.0/4.5 ინტეგრირებულ გარემოში - C#, Java, XAML, WPF, WCF, Oracle, MsSQL Server პაკეტების, Enterprise Architect, MsVisio და სხვა CASE ინსტრუმენტული საშუალებების გამოყენებით.

სისტემის მომხმარებლებისთვის შესაბამისი პროგრამული ინტერფეისების და ინსტრუქციების შემუშავება. პროექტის საპილოტო ვერსიის აგება, სადემონსტრაციო მაგალითების რეალიზაცია.

კვლევის ობიექტი. განაწილებული სისტემების პროგრამული უზრუნველყოფის ხარისხის მართვის მეთოდები ვირტუალური გარემოს პირობებში. პროგრამული სისტემის სასიცოცხლო ციკლის ეტაპები, კერძოდ მოთხოვნილებათა ობიექტ-ორიენტირებული ანალიზის, დაპროექტების, ტესტირების და დანერგვის ბიზნეს-პროცესები.

კვლევის მეთოდები. პროგრამული უზრუნველყოფის ბიზნეს-პროცესების ობიექტ-ორიენტირებული და პროცეს-ორიენტირებული მოდელირების მეთოდები, სერვის-ორიენტირებული არქიტექტურის

რეალიზაციის ინსტრუმენტული საშუალებები. საინფორმაციო სისტემების აგების ITIL და UML მეთოდოლოგიები და მათი CASE საშუალებები. მონაცემთა განაწილებული ბაზების თეორია. საფინანსო-საკრედიტო სისტემების თეორია, რისკების მართვის, მოდელირების და შეფასების მეთოდები. ექსპერტულ შეფასებათა მეთოდები, წარმოების ორგანიზაციულ-ტექნიკური დონის შეფასების თეორია.

მეცნიერული სიახლე. ნაშრომში განხილულია ვირტუალური გარემოს აგების პრინციპები და უპირატესობები ფიზიკურ სერვერებთან მიმართებაში. შემოთავაზებულია ვირტუალური გარემოს პრობლემების გადაწყვეტა თანამედროვე ინფორმაციული ტექნოლოგიების საფუძველზე. კერძოდ:

1. გამოკვლეულია Linux, Vmware და Oracle განაწილებული სისტემები ვირტუალურ გარემოში და მათი მართვის უპირატესობები ფიზიკურ სერვერებთან მიმართებაში;

2. ექსპერიმენტული მაგალითისა და დაკვირვების შედეგად შემოთავაზებულია რეკომენდაციები გარკვეული პრობლემების გადასაჭრელად. კერძოდ, გამოიკვეთა resourse pool-ის გამოყენების უპირატესობა Vapp-ის ნაცვლად, რადგან მასზე მანიპულირება უფრო მარტივი და მოსახერხებელია, ხარისხის თვალსაზრისით კი ორივე ერთნაირია;

3. განაწილებული სისტემებისთვის მონაცემთა დამუშავების ცენტრებში ინფორმაციის სწრაფი მიმოცვლის მახასიათებლების დადგენის მიზნით, შემუშავდა იმიტაციური მოდელი პეტრის ფერადი ქსელების გამოყენებით;

4. ჩატარებული ექსპერიმენტების შედეგების განზოგადებით მოხდა განაწილებული სისტემების პროგრამული უზრუნველყოფის ეფექტიანობის განსაზღვრა და მისი ხარისხის დონის ამაღლება ვირტუალური სერვერების გამოყენების საფუძველზე. ვირტუალურ გარემოში განსაკუთრებით სწრაფად ხდება ინფორმაციის მიმოცვლა სერვერსა და მყარ დისკოებს შორის, უფრო სწრაფად ხდება მონაცემთა აღდგენა (ბეკაპირება), უფრო სწრაფია ერთი სერვერიდან მეორე სერვერზე გადართვა, უფრო სწრაფია სერვერის

რესტარტი და უფრო მარტივია ვირტუალურ მანქანების მართვა ფიზიკურ სერვერებთან განსხვავებით.

შედეგების გამოყენების სფერო. დისერტაციის შედეგებს აქვს პრაქტიკული ღირებულება, ისინი შეიძლება გამოყენებულ იქნას ნებისმიერი ორგანიზაციული მართვის ბიზნეს-პროცესების მოდელირების, დაპროექტების და პროგრამული რეალიზაციის ამოცანების გადასაწყვეტად ახალი ინფორმაციული ტექნოლოგიებით.

ნაშრომის აპრობაცია: დისერტაციის ძირითადი შინაარსი მოხსენებული იყო ინფორმატიკისა და მართვის სისტემების ფაკულტეტის „მართვის ავტომატიზებული სისტემების (პროგრამული ინჟინერია)“ კოლეგიის სამეცნიერო სემინარების სხდომებზე და კოლოქვიუმებზე

პუბლიკაციები: დისერტაციის ძირითადი შედეგები გამოქვეყნებულია 6 სამეცნიერო ნაშრომში, რომლებიც ასევე მოცემულია ამავე დისერტაციაში.

ნაშრომის მოცულობა და სტრუქტურა: დისერტაციის სრული მოცულობა შეადგენს 120 ნაბეჭდ გვერდს; შედგება რეზიუმეს (ორ ენაზე), სარჩევის, შესავლის, 3 თავის და დასკვნისგან.

დისერტაციის მოკლე შინაარსი

დისერტაციის **პირველი თავი** ეთმობა პროგრამული უზრუნველყოფის ხარისხის მართვას ვირტუალური გარემოს პირობებში. რა არის პროგრამული ხარისხი, რა კრიტერიუმებით ფასდება. გადმოცემულია Vmware-ის დეტალები. განხილულია ვირტუალიზაციის დადებითი და უარყოფითი მხარეები. რისთვის და სად გამოიყენება, ვირტუალიზაციის სახეობები, აპარატურული ვირტულიზაციის შედარება პროგრამულთან. როგორია მისი წარმადობა და თვით ვირტუალური მანქანის შექმნა/კონფიგურაცია. თუ როგორ ყენდება ESX-ი და ა.შ.

ხარისხი ზოგადად სრულყოფილების გარკვეულ დონეს აღნიშნავს. ხარისხს შემდეგნაირად განმარტავენ: „პროდუქტის ან მომსახურების მახასიათებლების და თვისებების ერთობლიობა, რომელიც განაპირობებს მის შესაძლებლობას დააკმაყოფილოს მოთხოვნა”.

ბოლო რამდენიმე წლის განმავლობაში წარმოების/მომსახურების სფერო უკიდურესად შეიცვალა. მნიშვნელოვნად გაიზარდა კონკურენცია და კომპანიები ახალი გამოწვევის წინაშე აღმოჩნდა. კონკურენცია მხოლოდ ფასზე არ არის დამოკიდებული. ხშირად ბრძოლაში გადამწყვეტი ფაქტორი ხარისხს ენიჭება.

ხარისხის შეფასების ძირითადი პარამეტრები **ცხრ.1**

მატერიალური ნაკეთობა	მომსახურება
ძირითადი ფუნქციური მახასიათებლები	საიმედოობა
თავისებურებები	დამაჯერებლობა
საიმედოობა	უსაფრთხოება
შესაბამისობა	ხელმისაწვდომობა
მდგრადობა	ზრდილობა
ექსპლუატაციის ხანგრძლივობა	კომუნიკაბელურობა
ესთეტიკურობა	კომპეტენტურობა
აღდგენადობა	რეაგირება
აღსაქმელობა	კლიენტის შესწავლა
ფასეულობა	ხილული კომპონენტები

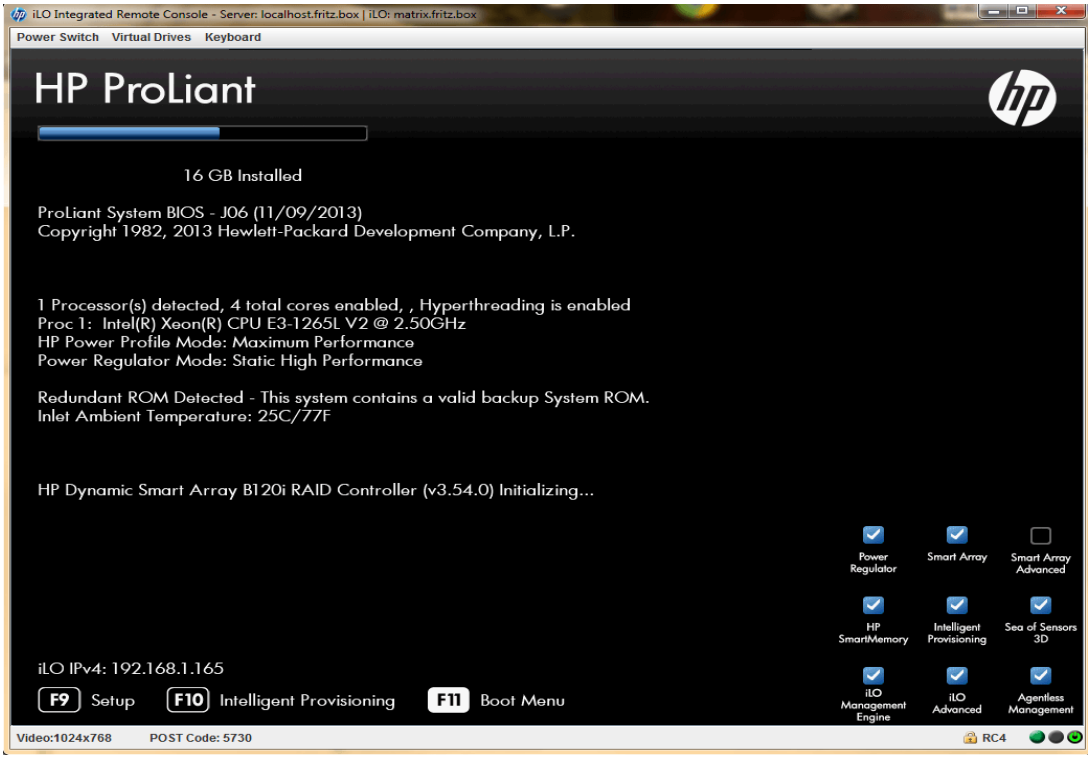
ვირტუალიზაციის მაგალითისთვის განვიხილოთ შემდეგი ამოცანა. გვაქვს Oracle-ს სამი ბაზა და შევხედოთ როგორ გამოიყურება ის ვირტუალურ მანქანაზე და ფიზიკურ მანქანებზე. ვირტუალურ მანქანის შემთხვევაში ეს სამივე ბაზა იქნება ერთ სერვერზე, ხოლო ფიზიკური მანქანის შემთხვევაში მისთვის საჭიროა სამი ფიზიკური სერვერი და ყველა სერვერზე დაყენებული იქნება მხოლოდ 1 ბაზა (ნახ.1).

ნახ.1

ხოლო ვირტუალიზაციის პირობებში ამის გადაწყვეტა უფრო მარტივია. მოვიგებთ ადგილსაც სასერვეროში და რესურსსაც, ასევე გარკვეულ ფულად დანახარჯებს. განვიხილოთ ეს საკითხი დეტალურად.

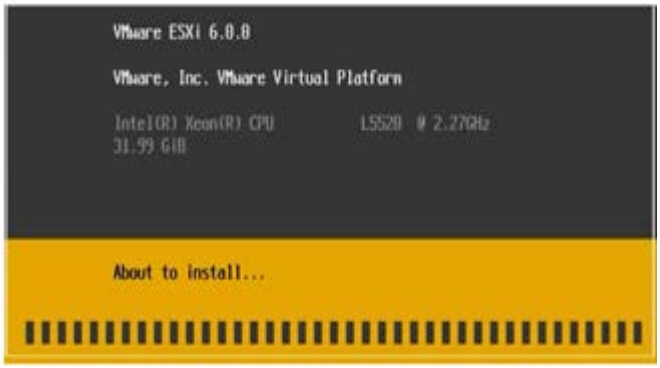
პირველ რიგში ვარჩევთ სერვერს თუ რა რესურსი აქვს. ამ ეტაპზე გამოვიყენოთ HP G7 სერვერები. შერჩევის შემდეგ საჭიროა მასზე დაყენდეს ESX-ი (ჩვენს შემთხვევაში დავაყენოთ VMware ESXi 6 ვერსია).

სერვერის გადატვირთვის შემდეგ ვირჩევთ თუ საიდან ჩაიტვირთოს (დისკო, ფლეშკა) ოპერაციული სისტემა (ნახ.2).



ნახ.2

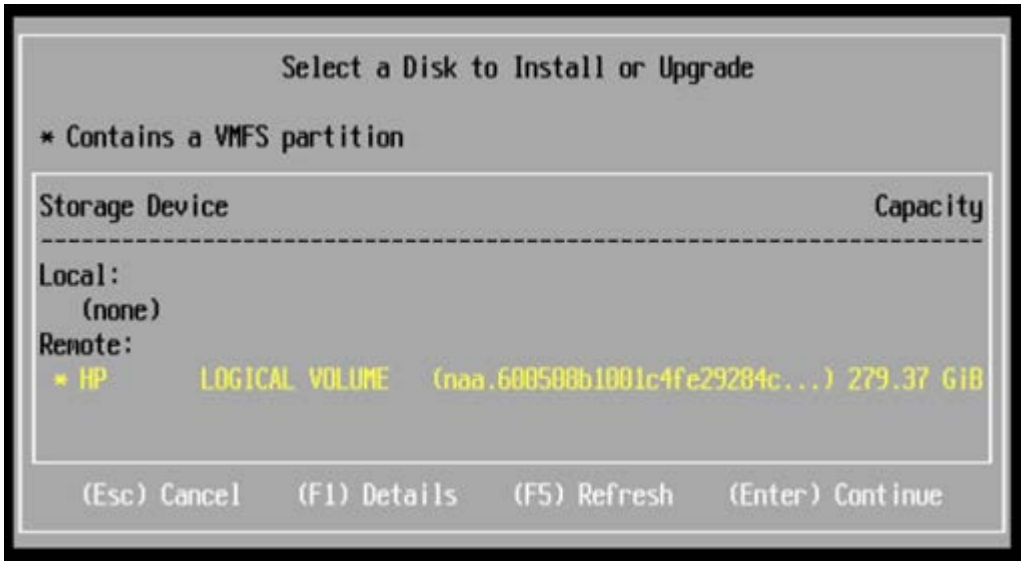
ამის შემდეგ ვუთითებთ ადგილს, თუ სად უნდა დაყენდეს ახალი VMware ESXi 6-ი. ჩვენ ავირჩიეთ ლოკალური დისკო (ნახ.3).



ნახ.3

ვაგრძელებთ ინსტალაციას. ყოველ შემდეგ ბიჯს ქვემოთ უწერია, თუ რა კლავიშზე დაჭერით ხდება გაგრძელება. ასევე მოგვთხოვს დავადოთ

პაროლი და შევიყვანოთ „იუზერი“ რომელიც აუცილებლად დაგვჭირდება შემდგომში (ნახ.4).



ნახ.4.

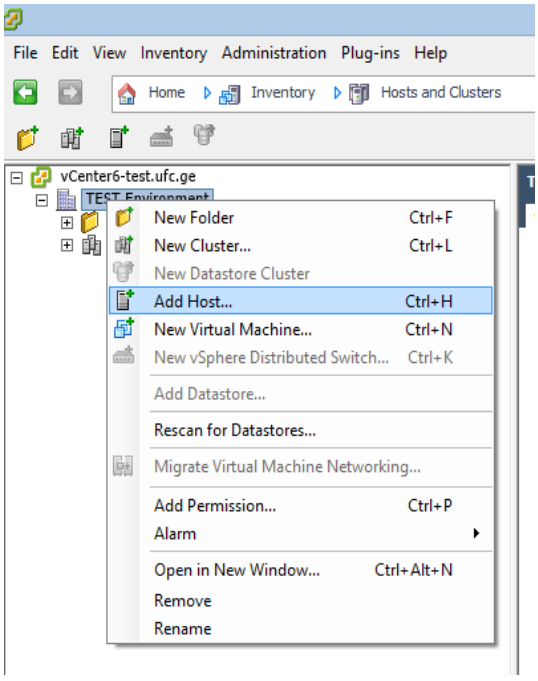
ამ პროცესის დასრულების შემდეგ ESX-ის ინსტალაცია დასრულებულია. ამის შემდეგ ხდება უკვე სერვერის კონფიგურირება (ნახ.5).

ნახ.5. სერვერის კონფიგურირება

სურათზე მოცემული პუნქტების შესაბამისად ავირჩევთ საჭირო მნიშვნელობებს. Network adapter- ში მივუთითებთ თუ რომელ ადაპტერშია მიერთებული ქსელი, რომელ VLAN-შია (მაგალითად, 135) ჩვენი სერვერი (თუ ქსელშია, რათქმა უნდა). IP configuration-ში ვწერთ მის IP მისამართს და

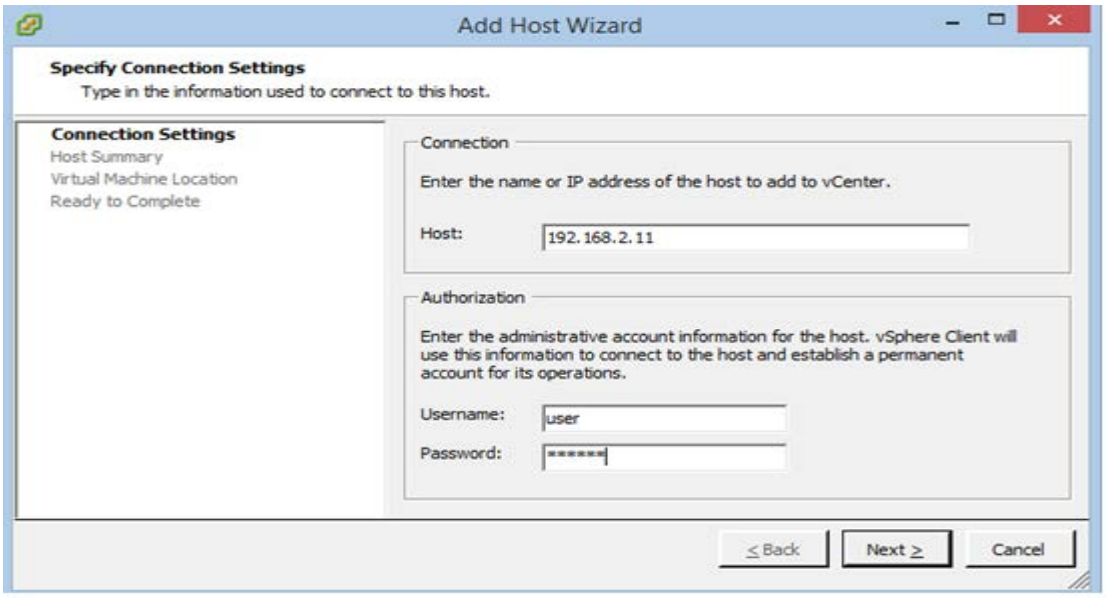
გეითვეის. ამის შემდეგ უკვე ვტვირთავთ მენეჯმენტ ქსელს (აქვე გვაქვს შესაძლებლობა test management network -ფუნქციონალით გავტესტოთ თუ გადის პინგი სხვა იგივე VLAN-ში მყოფ სერვერებზე. პინგი თუ გავიდა, მაშინ ყველაფერი რიგზეა). ძირითადი კონფიგურაციები უკვე თითქმის დასრულებულია. რჩება საჭირო ამ სერვერის გარედან მართვა და ასევე მეხსიერების მიბმა (ამ უკანასკნელზე დამატებით განვიხილავთ დეტალებს).

იმისთვის, რომ გარედან დავუკავშირდეთ აღნიშნულ სერვერს, საჭიროა ჩვენს კომპიუტერზე (ან საიდანაც გვინდა დაკავშირება) დავაყენოთ VMware vSphere Client-ი და დავამატოთ ჩვენი სერვერი, რაც ხდება შემდეგნაირად: შევდივართ vCenter-ში, მარჯვენა კლიკ და ვირჩევთ add_host. როგორც მე-6 ნახაზზეა მოცემული.



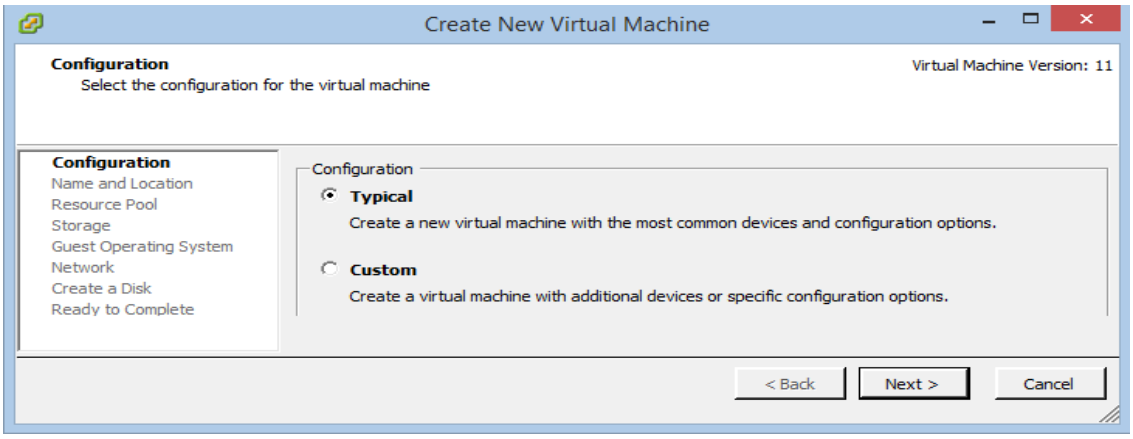
ნახ.6

ამის შემდეგ გამოდის ფანჯარა სადაც უნდა შევიყვანთ ჩვენი სერვერის IP მისამართს და user/pass (ნახ.7).



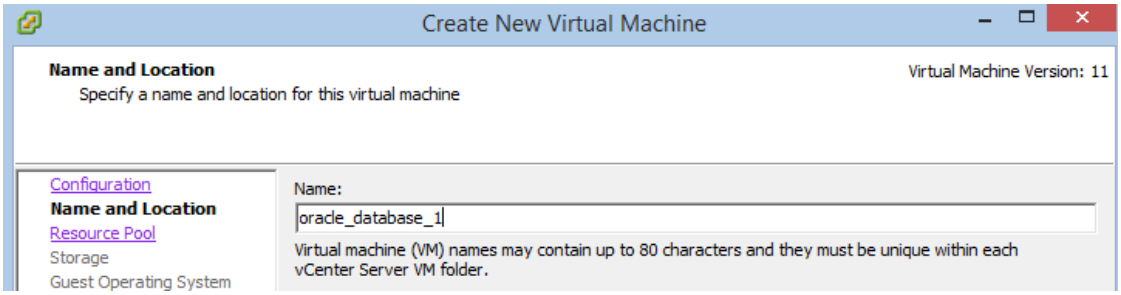
ნახ.7

Next-ებით მივებით და ვასრულებთ სერვერის დამატებას vCenter-ში. ჩამოთვლილი hardware და software პუნქტებში საჭიროების შემთხვევაში შეგვაქვს კორექტირებები. ამ ეტაპზე უკვე შეგვიძლია შევქმნათ ახალი ვირტუალური სერვერი, რომელიც შემდეგნაირად სრულდება: ვდგებით ჩვენ სერვერზე, მარჯვენა კლიკ და ვირჩევთ new virtual machine (ნახ.8).



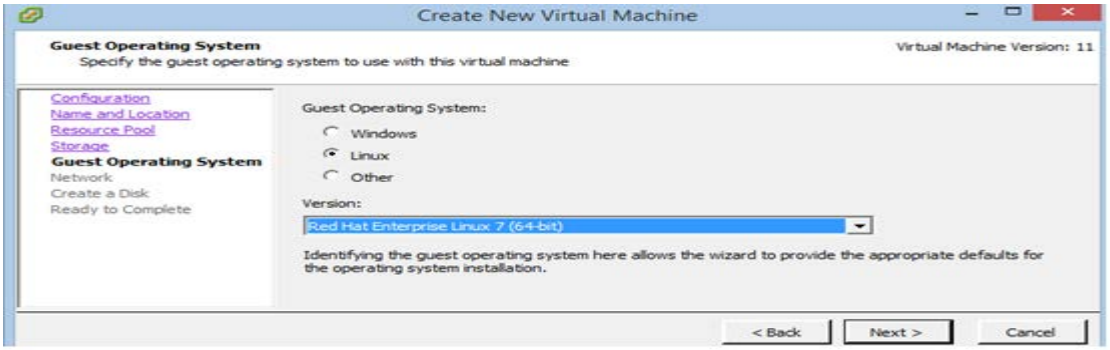
ნახ.8

ვირჩევ Typical და next. შეგვყავს ახალი ვირტუალური მანქანის სახელი და next (ნახ.9).



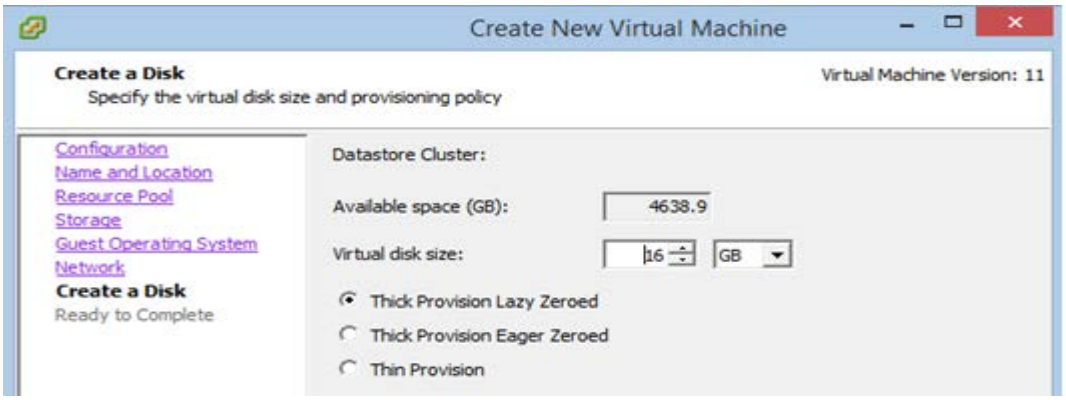
ნახ.9

ვირჩევთ რომელ ოპერაციულ სისტემისთვის ვქმნით ახალ სერვერს. ჩვენს შემთხვევაში მოვნიშნოთ ლინუქსი (ნახ.10).



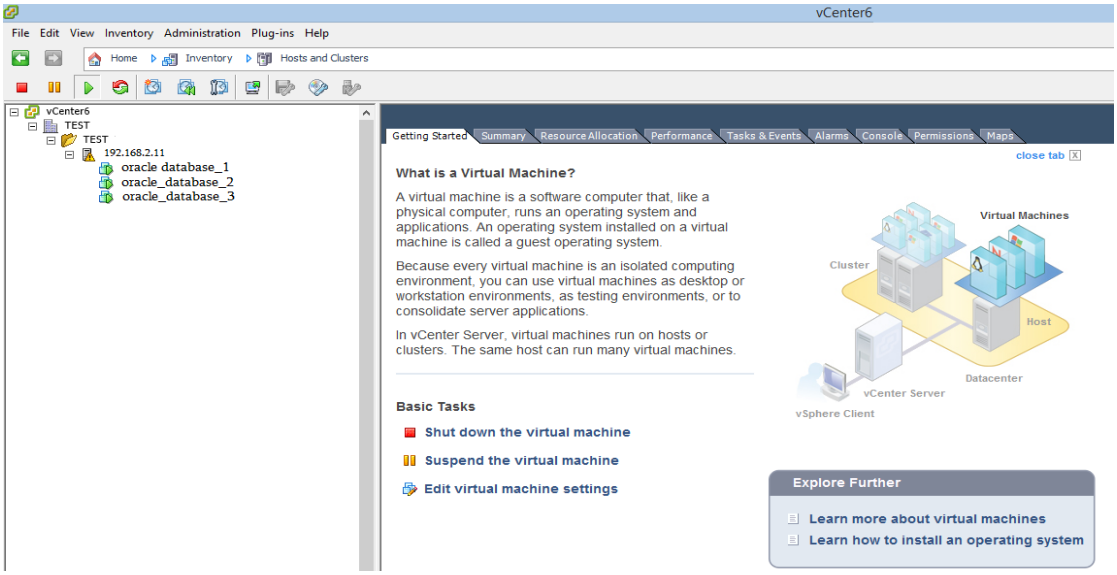
ნახ.10. Linux ოს არჩევა

ვირჩევთ სერვერის ზომას და finish (ნახ.11).



ნახ.11

ახალი ვირტუალური მანქანა მზადაა (ასე გავიმეორებთ სამივე სერვერის შემთხვევაში). ასეთი სახე ექნებათ ამ ვირტუალურ მანქანებს საბოლოოდ VMware vSphere Client-ში (ნახ.12).



ნახ.12

აქვე გვაქვს შესაძლებლობა დავაჯგუფოთ ბაზები და აპლიკაციები, რის საშუალებაც ასევე აქვს vCenter-ს, იქნება ეს resourse pool ან vApp. და თითოეულის შიგნით შევიტანოთ ჩვენი ვირტუალური მანქანები.

კომპანიაVMware 2007 წლიდან მთლიანად ცვლის გაყიდვების სტრატეგიას SMB სფეროში და იწყებს სპეციალიზირებულ პაკეტ-დამაჩქარებლების მოწოდებას (Acceleration Kits), რომლებიც შედგება თვით პლატფორმისაგან (ESX Serverან მსუბუქი ESX Server 3i), აგრეთვე Virtual Center ვირტუალიზაციის სერვერების Virtual Center მართვის საშუალებებს.

მთავარი ნაკლი ამ პაკეტების არის ის, რომ მათი ღირებულება მაღალია. ადრე პლატფორმა ESX Server აყენებდა საკმაოდ მკაცრ მოთხოვნებს დანადგარების მიმართ, ეხლა კი ჩანერგილი ჰიპერვიზორი ESX Server 3i ეხმარება SATA დისკებს და შეიძლება დაყენდეს ნოუთბუქებზეც კი. მალე დაიწყება სერვერების მოწოდება უკვე დაყენებული ESX3i პლატფორმით.

ამის გარდა დღესდღეობით კომპანია VMware ატარებს VMware Server 2.0 უფასო ვირტუალიზაციის პლატფორმის ბეტა-ტესტირებებს, რომელიც არის კარგი გადაწყვეტილება, რათა გამოიცადოს ვირტუალიზაცია მუშაობაში და ჩაატაროს ვირტუალური სერვერების საცდელი ექსპლუატაცია.

VMware ESX \ VMware ESXi

ESX: ESX შედგება 2 ძირითადი კომპონენტისაგან - ჰიპერვიზორი და Linux. ჰიპერვიზორი - ეს არის პროგრამისტების მიერ დაწერილი VMware კომპონენტი, რომელიც „აკეთებს“ ვირტუალიზაციას. მას აგრეთვე უწოდებენ ბირთვს, "vmkernel".

Linux - ეს არის დანაწევრებული Red Hat Enterprise Linux 3. დანაწევრება ნიშნავს, რომ ამოღებულია ყველა არა საჭირო ESX კომპონენტი, მაგ. არ არის ftp სერვერი. ეს ლინუქსი გამოიყენება ბრძანების ველის ლოკალური ინტერფეისის მისაღებად. ერთადერთი, რაც ჩემი აზრითრაც შეიძლება გაკეთდეს, ეს არის მანქანის მონიტორინგის და ბექაპის აგენტების ლინუქსში ინსტალაცია.

ESXi: შედგება 2 ძირითადი კომპონენტისაგან - ჰიპერვიზორი და Linux, მაგრამ ლინუქსი ძალიან პატარაა და პრაქტიკულად ლოკალურად ვერავითარ ფუნქციას ვერ ასრულებს. სამაგიეროდ პატარა ლინუქსის გამო მთელი ESXi ეტევა 30 მეგაბაიტში და არა 1,5 გიგაბაიტში, ისევე როგორც ESX.

რეზუმე: განსხვავება - ფუნქციონალური ვირტუალიზაციის მხრიდან, სულერთია გამოიყენება ESX თუ ESXi.

პირველს ექნება აგრეთვე ლინუქსი - მასში შეიძლება რაიმე დავაინსტალიროთ და შეგვიძლია ვიმუშაოთ ჩვეული ბრძნების სტრიქონით.

მაგრამ „i“ ვერსიით - შეიძლება ვიყიდოთ სერვერი, სადაც ეს ფლეშკა ESXi-ით უკვე ჩაყენებულია და ასე რომ შეგვიძლია გამოვტოვოთ ინსტალაციის ეტაპი.

- შეიძლება ჩავწეროთ ფლეშკაზე და სერვერი ჩავტვირთოთ ამ ფლეშკიდან (ეს მეთოდი, რა თქმა უნდა, ოფიციალურად არაა რეკომენდირებული);

- შეიძლება ორგანიზება გავუკეთოთ PXE ჩატვირთვას - მაგრამ ამ შემთხვევაშიც ასეთი კონფიგურაცია არ არის სასურველი;

- ლოკალურად არ შეიძლება არაფრის დაინსტალირება და არც არის საჭირო;

- წვდომა ლოკალურ ბრძანების ველთან შესაძლებელია მხოლოდ არაოფიციალურად. მაგრამ არის მოშორებული ბრძანების ველი, რომელსაც ეწოდება RCLI (Remore Command Line Interface), ე.ი. ეს პუნქტი მინუსად არ ჩაითვლება.

ოფიციალური ჩამონათვალი, რომლებიც არ ნარჩუნდება ESXi 3.5.Update 2, მაგრამ ნარჩუნდება ESXi 6.5.Update 2-Differences in Supported Networking Features Between ESX Server 6.5 and ESX Server 6i.

დაყენება და კონფიგურირება VMware ESXi

ESXi Server-ის საფუძველი

ESXi-ის უფასო ვერსია შეიძლება დაყენებული იყოს სერვერზე (ჩატვირთავა ფლეშკიდან) ან შეიძლება დაყენდეს უკვე არსებულ სერვერზე Installable ვერსიის გამოყენებით. ESXi-ის უფასო ვერსიაში შედის მხოლოდ VMFS და vSMP მხარდაჭერა დამატებითი შესაძლებლობების გარეშე. ESXi-ის დაყენებას სჭირდება დაახლოებით 5GB დისკზე. დარჩენილი ადგილი ფორმატირდება როგორც VMFS პარტიცია. თვითონ ჰიპერვიზორი გამოიყენება მხოლოდ 32Mb, დამატებით ადგილს იკავებს VMware tool, სვაპი და ბირთვის დამპი.

არსებობს ESXi-ის 4 ვერსია: ESXibase, რომელიც მოიცავს VMFS და vSMP; ESXiFoundation, სადაც დამატებულია Virtual Center Agent, Update Manager და Consolidated Backup; ESXi Standard -დამატებულია High Availability; და ESXi Enterprise სადაც დამატებულია VMotion, Storage VMotion, DRS და DPM ფუნქციონალური შესაძლებლობები.

ESXi-ის საბაზო ვერსია ვერ იმართება VirtualCentre-ის მეშვეობით, რადგან მასში არ შედის VirtualCentre-ის აგენტი. ESXi-ის საინსტალაციო ვერსიას აქვს მოწყობილობების შესაბამისობის საკუთარი სია (HCL). თუმცა ESXi-ის დაყენება შესაძლებელია სხვა მოწყობილობაზე, ზემოაღნიშნული სია მოიცავს ოფიციალურ მოწყობილობებს, რომლზედაც VMware-ს აქვს წვდომა.

ESXi-ის ფაიერვოლი ადვილი სამართავია, რადგან არ არსებობს მართვის კონსოლის დაცვის აუცილებლობა. ლოდინის რეჟიმში მასზე დასაშვებია მხოლოდ ორი სახის მიერთების განხორციელება. ESXi-ს, ბევრი რედაქციისგან გასხვავებით, არ გააჩნია ვებ-ინტერფეისი. ჰოსთის მოწყობილობის მონიტორინგი ხორციელდება ESXi-ში, CIM პროვაიდერის მეშვეობით და მისი დანახვა შესაძლებელია VI კლიენტში.

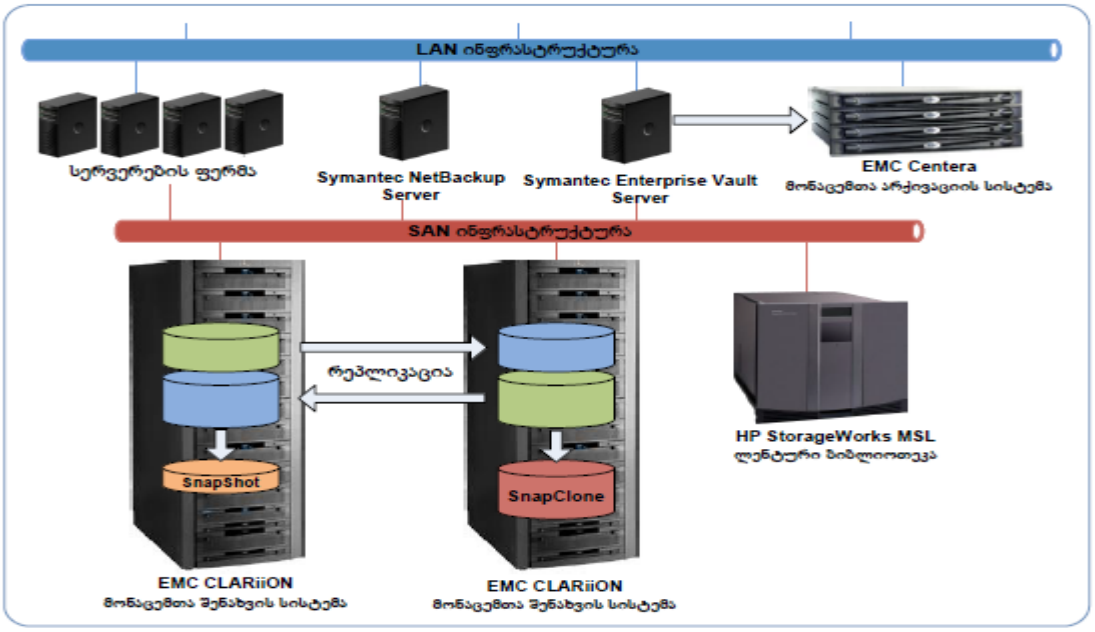
ESXi-ის უფასო ვერსიებიც უნდა იყოს ლიცენზირებული. ლიცენზიის მისაღებად მომხმარებელი უნდა დარეგისტრირდეს Vmware-ის ვებ-გვერდზე და ის ელექტრონულ ფოსტაზე მიიღებს ლიცენზიის გასაღებს.

ESX-გან განსხვავებით, ESXi ავლენს ძალიან ცოტა საინსტალაციო შეტყობინებას, რაც დაგეხმარებოდათ სერვერის კონფიგურირებაში. მაგალითად, ქსელის კონფიგურაცია ხდება ინსტალაციის შემდეგ (ნახ.13).

დისერტაციაში განხილულია ასევე კომპიუტერულ ქსელები. მათი უპირატესობა სწორედ იმაშია, რომ ამ დროს კეთდება თანხების დაზოგვა (როგორც ზემოთ აღვნიშნეთ, არაა საჭირო თითოეული კომპიუტერისთვის სკანერის, პრინტერს ან სხვა რომელიმე აპარატის ყიდვა).

ნაშრომის მეორე თავში მოცემულია პროგრამული უზრუნველყოფის სასიცოცხლო ციკლის მართვის გუნდის როლებისა და ფუნქციების აღწერა. აგებულია ფუნქციონალური მოთხოვნილებების განსაზღვრის UseCase, Activity დიაგრამები.

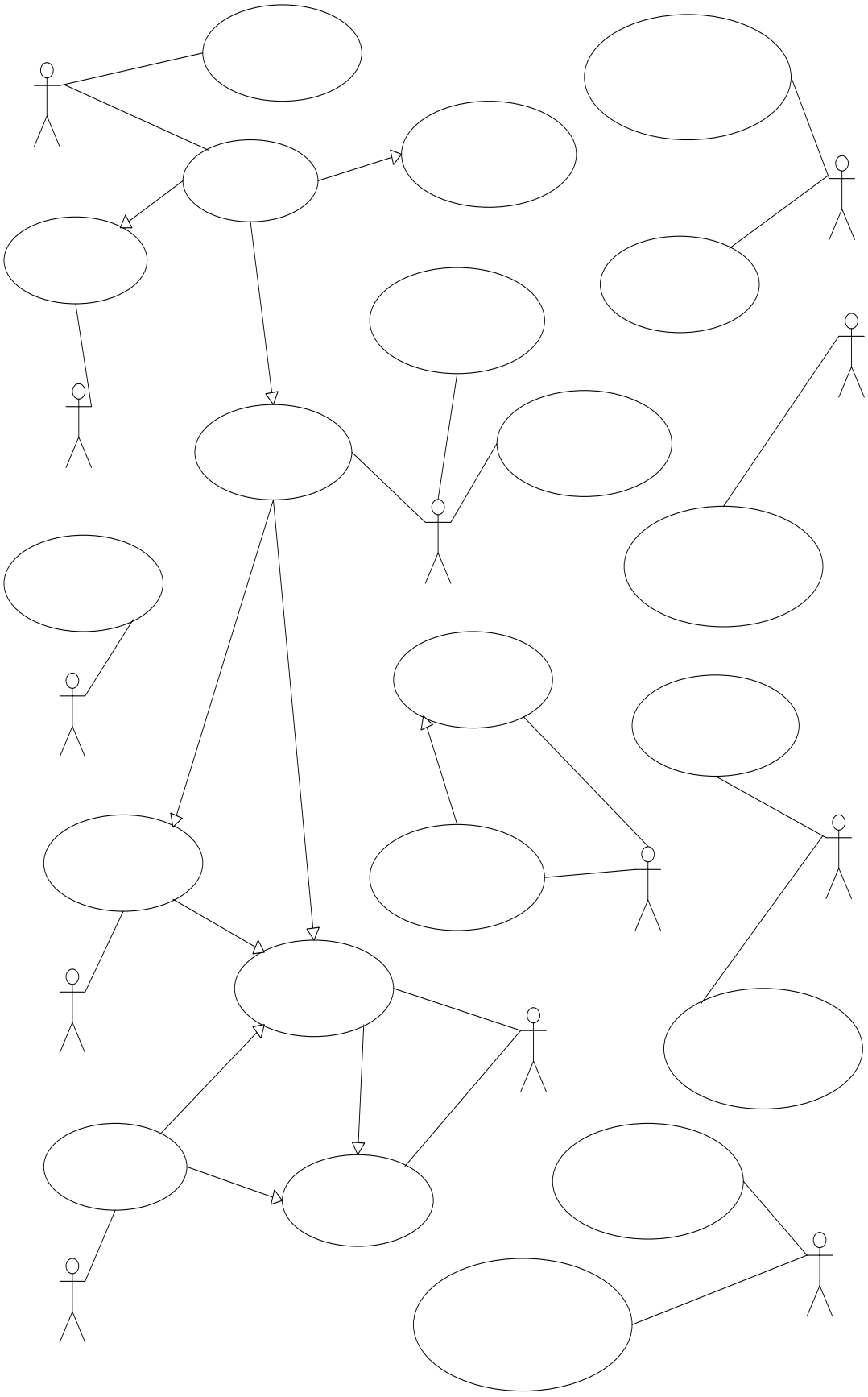
პროგრამული სისტემების პროექტის მენეჯერისათვის მნიშვნელოვანია პროდუქტის სასიცოცხლო ციკლის ბიზნეს პროცესების დეტალური ანალიზი, მის ცალკეულ ეტაპებზე ეფექტური მართვის განსახორციელებლად.



ნახ.13

მე-14 ნახაზზე მოცემულია პროგრამული პროექტების შესრულების ერთიანი პროცესის ზოგადი მოდელი, აგებული UML-ის UseCase დიაგრამით. დისერტაციაში შემუშავებულია ასევე შესაბამისი ბიზნეს-პროცესების და ბიზნეს-წესების ამსახველი აქტიუროზათა დიაგრამა. აღწერილი ზოგადი მოდელები მენეჯერის მიერ ადაპტირება კონკრეტული პროექტისათვის და ღებულობს კერძო სახეს. მთლიანი სისტემის ბიზნეს-პროცესების ანალიზი საფუძველია სისტემის არქიტექტურის და საერთოდ, IT-ინფრასტრუქტურის დასადგენად.

ნაშრომში წარმოდგენილია აგრეთვე მონაცემთა დამუშავების ცენტრებში სერვერებსა და კლიენტებს შორის ინფორმაციის გაცვლის პროცესის იმიტაციური მოდელის აგების ამოცანა პეტრის ფერადი ქსელების (CPN – Coloured Petr Network) საფუძველზე, რაც ექსპერიმენტებისთვის შესანიშნავი ინსტრუმენტული საშუალებაა.



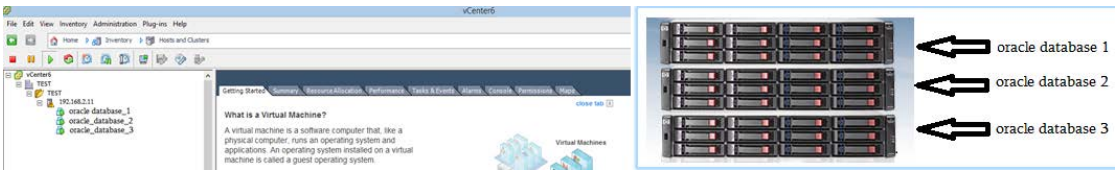
ნახ.14

პროგრამული სისტემების სასიცოცხლო ციკლის, კერძოდ განტერის „ფაზა-ფუნქციების“ მოდელის საფუძველზე

მესამე თავში განხილულია ის შედეგები რომლებიც მივიღეთ გარკვეული ექსპერიმენტების და ცდების შედეგად ვირტუალურ გარემოსთან მიმართებაში. მაგალითისთვის განვიხილოთ შემდეგი ამოცანა. გვაქვს Oracle-ს სამი ბაზა და შევხედოთ როგორ გამოიყურება ის ვირტუალურ მანქანაზე და ფიზიკურ მანქანებზე. ვირტუალურ მანქანის შემთხვევაში ეს სამივე ბაზა იქნება ერთ სერვერზე, ხოლო ფიზიკური მანქანის შემთხვევაში მისთვის საჭიროა სამი ფიზიკური სერვერი და ყველა სერვერზე დაყენებული იქნება მხოლოდ 1 ბაზა (ნახ.1).

ვირტუალიზაციის პირობებში მივიღთ ბევრად მეტი წარმადობა ფიზიკურ სერვერთან შედარებით, დროშიც და ფინანსურადაც. ზემოთ აღწერილი ამოცანის გადასაწყვეტად საჭიროა შევიძინოთ 3 G7 სერვერი, რომელშიც გადასახდელია 3X თანხა. ვირტუალური გარემოში ამის გადასაწყვეტად ვყიდულობთ 1 G7 სერვერს რომელიც ღირს 1X თანხა, ვაძლევთ ცოტა მეტ ოპერატიულ მეხსიერებას (რადგან მასზე 3 ბაზა უნდა დაყენდეს და შესაძლებელია ის რესურსი, რაც მოჰყვა ამ სერვერს ოპერატიულ მეხსიერებაში, შესაძლებელია არ იყოს საკმარისი), ვაყენებთ მასზე ESX6.5-ს (როგორც ზემოთაა ნაჩვენები) და ვქმნით 3 ვირტუალურ მანქანას. ყველა ეს ფინანსური დანახარჯი არ ცდება 1.5X თანხას.

ასევე მოვიგეთ ადგილი სასერვეროში, რადგან 3 სერვერის მაგივრად გვაქვს 1 სერვერი რომელიც იმავე საქმეს აკეთებს რაც მისი წინამორბედი 3 სერვერი(ნახ.16).



ნახ.17

მოგეხსენებათ გარკვეული დროის შემდეგ გამოდის ახალი ვერსიები(linux,Oracle და ა.შ), რომელსაც განახლება სჭირდება. იმისათვის

რომ ლინუქსი ან ორაკლი განახლდეს სერვერზე საჭიროა მთლიანი სერვერის გადატვირთვა. ამაშიც მოვიგეთ დრო და 3 ჯერ ნაკლები დრო მოანდომა ვირტუალურ მანქანამ ვიდრე ფიზიკურ სერვერმა.

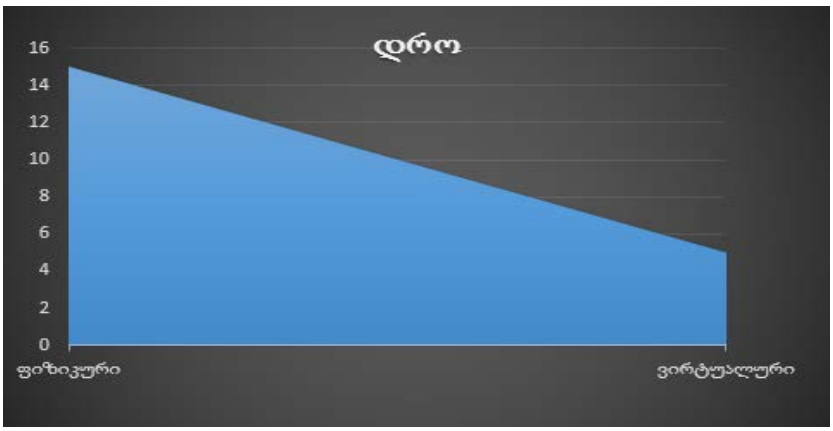
მონაცემთა ბეკაპირებაც უფრო გაუმჯობესდა და დაახლოებით 2 საათიდან ფიზიკურის შემთხვევაში ახლა ჭირდება 50-55 წუთი.

ასევე დიდი წარმადობა აჩვენა მონაცემების მყარ დისკე ჩაწერა/წაკითხვაში. და კიდევ ერთი პრობლემის შემთხვევაში თუ ვირტუალურ მანქანას რაიმე პრობლემა შეექმნა და გაჩერდა ავარიულად, სისტემა ავტომატურად რთავს მას მეორე, სათადარიგო მანქანაზე.

იმ შემთხვევაში თუ დაგვჭირდა სადმე იგივე მანქანის შექმნა (მაგ., სატესტო გარემოსთვის), არ გვიწევს ზემოთ ჩამოთვლილი ბიჯების თავიდან გავლა. VMware აქვს ასეთი ფუნქცია „clone“ რისი მეშვეობითაც რამდენიმე წუთში გვაქვს იგივე კონფიგურაციის და მონაცემების სერვერი, რაც რეალურ სერვერს. ფიზიკურის შემთხვევაში ასეთ რამეს ვერ შევძლებთ იგივე დროში.

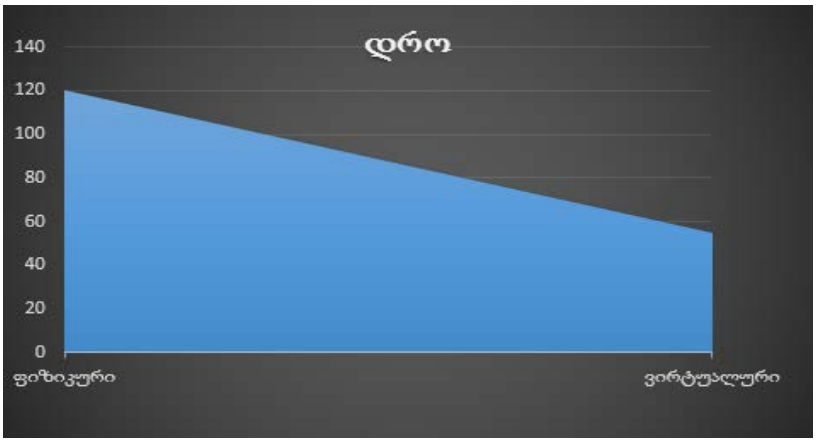
უკეთესად მუშაობს მონაცემთა დუბლირება/რეპლიკაცია.

ჩავატარეთ ექსპერიმენტები ფიზიკური სერვერიდან ვიტრუალურზე გადასვლელად. ზემოთ ვახსენე გადატვირთვის დრო და ნახაზზეც ჩანს თუ რამდენად შემცირდა იგი (ნახ.18).



ნახ.18

ქვემოთ ნახაზზე ნაჩვენებია მონაცემთა ბეკაპირების დრო (ნახ.19). ესეც ერთ ერთი მნიშვნელოვანი მომენტია, რადგან ბეკაპირების დროს სისტემა საკმაოდ იტვირთება და არ უნდა მოხდეს მისი ბეკაპირება დატვირთულ დროს(ძირითადად სერვერების ბეკაპირება ხდება ღამის საათებში).



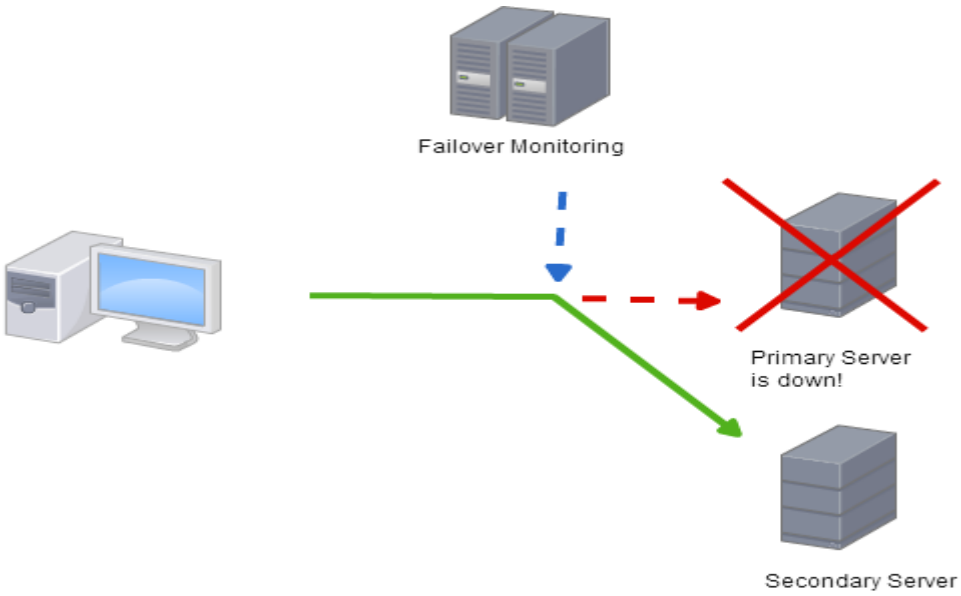
ნახ.19

ქვემოთ ნახაზზე ნათლად ჩანს თუ რა მინიმალური დრო სჭირდება მონაცემთა მყარ დისკზე ჩაწერა/წაკითხვას (ნახ.20). ამ შემთხვევაში მყარი დისკის სისწრაფესაც გააჩნია, რათქმაუნდა. ლაპარაკია ერთიდაიგივე მყარ დისკზე ვირტუალური/ფიზიკური გარემოს შემთხვევაში.

```
[oracle@fronttest3 ~]$ iostat -xmn 1
Linux 2.6.32-504.8.1.el6.x86_64 (fronttest3)    06/04/16        _x86_64_        (2 CPU)

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await  svctm  %util
sda               1.13     4.84    2.22    1.96     0.06     0.03    42.06     0.02    4.20   2.49   1.04
sdb               0.36    39.19   24.33   30.70     0.41     0.27    25.58     0.12    2.16   1.38   7.61
sdc               0.11    64.07   53.04   42.26     2.84     0.42    70.06     0.05    0.50   1.78  17.00
dm-0              0.00     0.00   27.43   74.16     0.47     0.29    15.30     0.28    2.71   0.82   8.31
dm-1              0.00     0.00    0.04    1.61     0.00     0.01    10.01     0.19  116.25   0.03   0.00
dm-2              0.00     0.00    0.00    0.01     0.00     0.00    11.81     0.00   57.82   1.41   0.00
dm-3              0.00     0.00    0.11    0.12     0.00     0.00     8.67     0.00    7.02   1.41   0.03

Filesystem:             rMB_nor/s    wMB_nor/s    rMB_dir/s    wMB_dir/s    rMB_svr/s    wMB_svr/s     ops/s    rops/s    wops/s

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await  svctm  %util
sda               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sdb               0.00     0.00    1.00    2.00     0.00     0.01     8.00     0.00    0.67   0.67   0.20
sdc               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-0              0.00     0.00    1.00    2.00     0.00     0.01     8.00     0.00    0.67   0.67   0.20
dm-1              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-2              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-3              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

Filesystem:             rMB_nor/s    wMB_nor/s    rMB_dir/s    wMB_dir/s    rMB_svr/s    wMB_svr/s     ops/s    rops/s    wops/s

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await  svctm  %util
sda               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sdb               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sdc               0.00     1.00    0.00    1.00     0.00     0.01    16.00     0.00    1.00   1.00   0.10
dm-0              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-1              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-2              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-3              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

Filesystem:             rMB_nor/s    wMB_nor/s    rMB_dir/s    wMB_dir/s    rMB_svr/s    wMB_svr/s     ops/s    rops/s    wops/s

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await  svctm  %util
sda               0.00     6.00    0.00    3.00     0.00     0.04    24.00     0.00    1.00   1.00   0.30
sdb               0.00    15.00   17.00   40.00     0.07     0.21    10.11     0.04    0.67   0.67   3.80
sdc               0.00     5.00    0.00    3.00     0.00     0.03    21.33     0.00    1.33   1.33   0.40
dm-0              0.00     0.00   17.00   64.00     0.07     0.25     8.00     0.06    0.69   0.51   4.10
dm-1              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-2              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-3              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

Filesystem:             rMB_nor/s    wMB_nor/s    rMB_dir/s    wMB_dir/s    rMB_svr/s    wMB_svr/s     ops/s    rops/s    wops/s

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await  svctm  %util
sda               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sdb               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sdc               0.00     3.00    0.00    2.00     0.00     0.02    20.00     0.00    1.50   1.50   0.30
dm-0              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-1              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-2              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
dm-3              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

Filesystem:             rMB_nor/s    wMB_nor/s    rMB_dir/s    wMB_dir/s    rMB_svr/s    wMB_svr/s     ops/s    rops/s    wops/s

^C
[oracle@fronttest3 ~]$
```

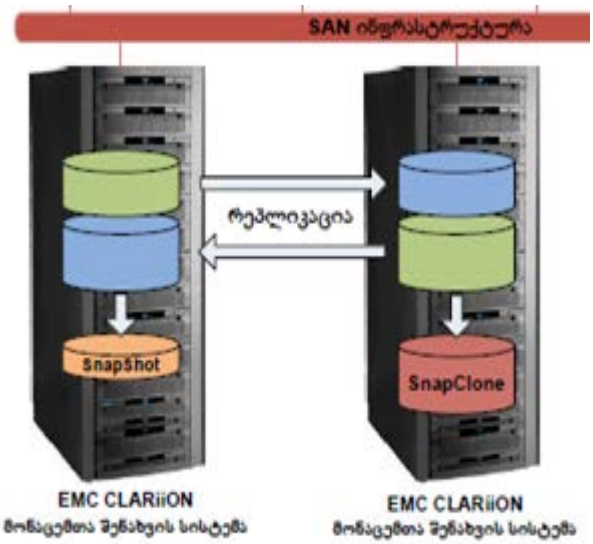
ნახ.20

21-ე ნახაზზე განხილულია ავარიულად გათიშვის შემთხვევაში სერვერების გადართვის პროცესი.



ნახ.21

22-ე ნახაზზე კი მოცემულია თუ როგორ მუშაობს რეპლიკაცია ორ სერვერს შორის.



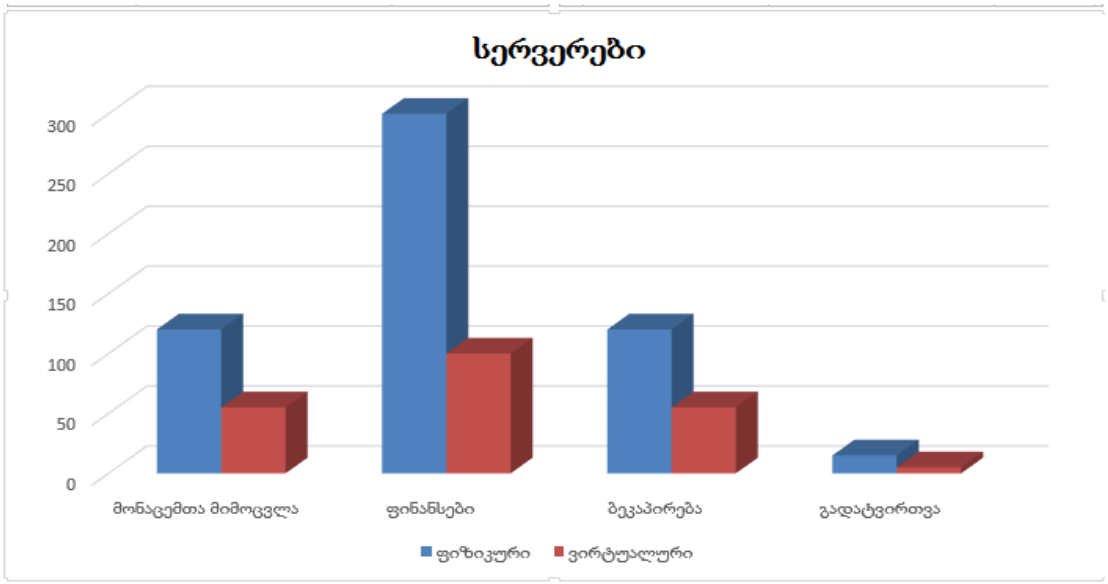
ნახ.22

23-ე ნახაზზე მოცემულია ორაკლის სერვერი მუშა მდგომარეობაში დატვირთულ დროს, თუმცა ეს დატვირთვა პრობლემას არ წარმოადგენს.

24-ე ნახაზზე ასახულია ვირტუალური გარემოს დროის და დანახარჯების შეფარდება ფიზიკურთან შედარებით.

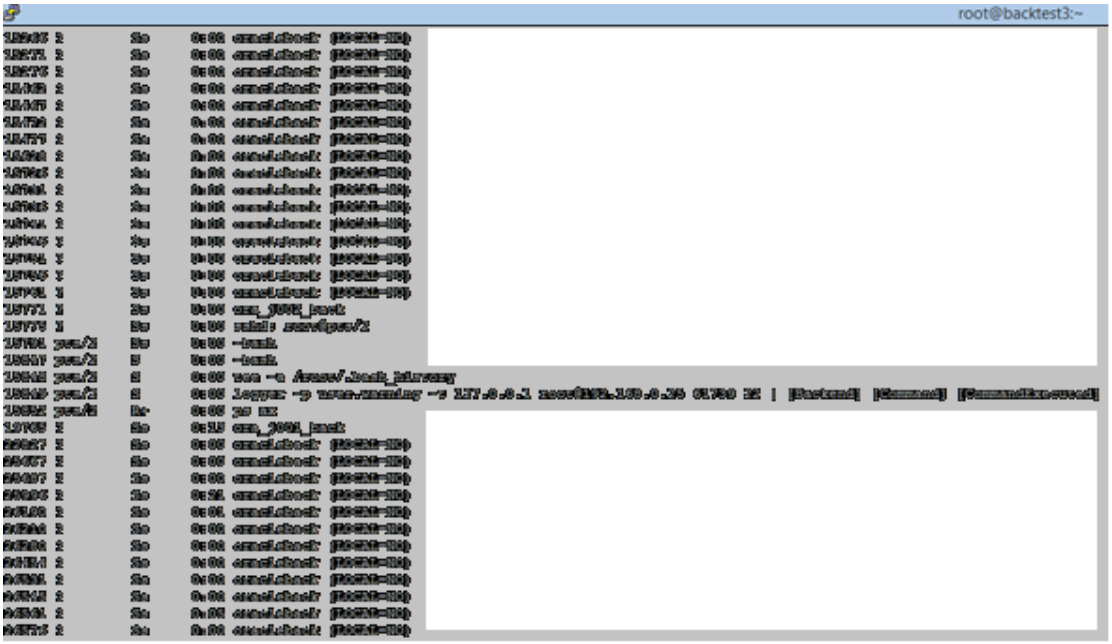
```
top - 17:46:24 up 149 days, 21:03,  1 user,  load average: 1.88, 1.58, 1.24
Tasks: 673 total,   1 running, 672 sleeping,   0 stopped,   0 zombie
Cpu(s): 22.2%us,  3.0%sy,  0.0%ni, 31.3%id, 43.3%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:   8061572k total,  7929944k used,   131628k free,     2392k buffers
Swap: 20971516k total,  1665500k used, 19306016k free,  4162956k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1543 oracle    20   0 5370m 214m 203m D 15.7  2.7   0:04.83 oracle
  940 oracle    20   0 5359m 534m 531m S  8.8  6.8   0:06.54 oracle
  722 oracle    20   0 5360m 1.2g 1.2g S  7.2 15.4   0:21.90 oracle
 3539 oracle    20   0 5361m 177m 172m S  4.6  2.2   6:03.64 oracle
 7038 oracle    -2   0 5357m 1900 1784 S  2.0  0.0   2531:51 oracle
 1057 oracle    20   0 5360m 591m 588m S  1.3  7.5   0:05.88 oracle
 1251 oracle    20   0 5358m 525m 522m S  1.3  6.7   0:02.15 oracle
 1414 oracle    20   0 5359m 188m 185m S  1.3  2.4   0:00.73 oracle
 1512 oracle    20   0 15440 1704  936 R  1.3  0.0   0:00.42 top
  351 oracle    20   0 5359m 525m 522m S  1.0  6.7   0:02.63 oracle
 1385 oracle    20   0 5359m 328m 325m S  1.0  4.2   0:01.07 oracle
 1501 oracle    20   0 5359m 349m 345m S  1.0  4.4   0:00.87 oracle
  816 oracle    20   0 5359m 613m 610m S  0.7  7.8   0:07.94 oracle
 1377 oracle    20   0 5359m 265m 262m S  0.7  3.4   0:01.90 oracle
 1389 oracle    20   0 5359m 311m 308m S  0.7  4.0   0:00.80 oracle
 1547 oracle    20   0 5358m 167m 164m S  0.7  2.1   0:00.19 oracle
 7064 oracle    20   0 5368m 114m 107m S  0.7  1.5 593:28.36 oracle
17881 oracle    20   0 5404m 2.1g 2.1g S  0.7 27.1 388:22.13 oracle
   23 root      20   0     0    0    0 S  0.3  0.0  99:10.05 kblockd/1
   40 root      20   0     0    0    0 S  0.3  0.0   2232:32 kswapd0
  914 oracle    20   0 5359m 535m 532m S  0.3  6.8   0:03.72 oracle
 1337 oracle    20   0 5359m 526m 523m S  0.3  6.7   0:01.55 oracle
 7074 oracle    20   0 5368m 1.6g 1.5g S  0.3 20.2  93:43.31 oracle
 7095 oracle    20   0 5365m 1.6g 1.5g S  0.3 20.2  90:58.56 oracle
 7099 oracle    20   0 5372m 137m 137m S  0.3  1.7 833:11.22 oracle
10629 oracle    20   0 5357m  13m  11m S  0.3  0.2   1:33.92 oracle
17826 oracle    20   0 5384m 1.5g 1.5g S  0.3 19.1  89:57.68 oracle
17885 oracle    20   0 5368m 334m 327m S  0.3  4.3 209:07.66 oracle
25564 oracle    20   0 5362m 1.6g 1.6g S  0.3 20.7 343:09.39 oracle
    1 root      20   0 19348  600  372 S  0.0  0.0  29:19.63 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.12 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   1:00.66 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   8:59.32 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:49.69 watchdog/0
    7 root      RT   0     0    0    0 S  0.0  0.0   0:55.53 migration/1
    8 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/1
    9 root      20   0     0    0    0 S  0.0  0.0   8:33.54 ksoftirqd/1
   10 root      RT   0     0    0    0 S  0.0  0.0   0:27.01 watchdog/1
   11 root      20   0     0    0    0 S  0.0  0.0  11:23.38 events/0
   12 root      20   0     0    0    0 S  0.0  0.0  14:40.03 events/1
```

ნახ.23. Oracle სერვერის დატვირთვის მაჩვენებლები



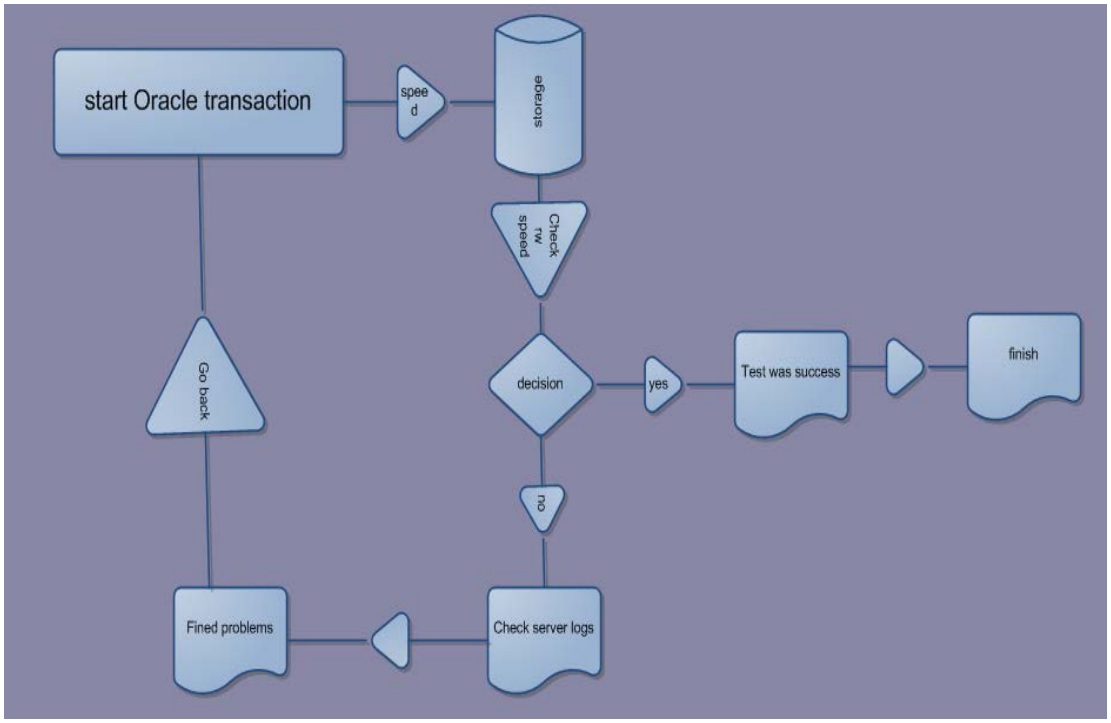
ნახ.24. ფიზიკური და ვირტუალური სერვერების შედარება

ამავე თავში მოცემულია ORACLE სისტემის არქიტექტურა და ძირითადი ობიექტები. მისი მუშაობის პრინციპები და დეტალები ვირტულური გარემოს პირობებში და უშუალოდ როგორ უნდა მოხდეს მისი ფიზიკური სერვერიდან ვირტუალურზე გადატანა (ნახ.25).



ნახ.25

ასევე შესაძლებელია გადამოწმდეს რამდენად დატვირთულია სერვერი. 26-ე ნახაზზე მოცემულია ვირტუალიზაციის პროცესის ეტაპები, რომელიც ციკლურად მიმდინარეობს საბოლოო შედეგის მიღებამდე.



ნახ.27. ვირტუალიზაციის პროცესის ეტაპები

„გავირტუალიზების“ შემდეგ ვაკვირდებით მუშაობისუნარიანობას და სისწრაფეს. თუ მისაღებია ეს მაჩვენებლები, მაშინ ვტოვებთ ვირტუალიზაციის ასეთ კონფიგურაციას. თუ არა, ვამოწმებთ სერვერის ლოგებს, ვაანალიზებთ შეცდომებს. თუ შესაცვლელია კონფიგურაცია ვცვლით მას და ვიწყებთ თავიდან საძიები მახასიათებლების ანგარიშს.

დასკვნა

1. განხილულია განაწილებული სისტემების პროგრამული უზრუნველყოფის ხარისხის მართვის პრობლემები და ამოცანები ტრადიციული და ვირტუალური გარემოს პირობებში. გაანალიზებული და კლასიფიცირებულია დღეისათვის არსებული სერევერული ტექნოლოგიები მართვის ავტომატიზებულ სისტემებში;

2. ახალი ტექნოლოგიების საფუძველზე ჩამოალიბებულია ვირტუალური გარემოს აგების პრინციპები და გამოვლენილია უპირატესობები (ტრადიციულ) ფიზიკურ სერვერებთან მიმართებაში;

3. ექსპერიმენტული ცდებისა და დაკვირვების შედეგად შემოთავაზებულია რეკომენდაციები არსებული პრობლემების გადასაჭრელად. კერძოდ, უპირატესობა ენიჭება ვირტუალურ სერვერებს, რადგან მათ საფუძველზე უფრო სწრაფად ხდება ინფორმაციის მიმოცვლა სერვერსა და მყარ დისკოებს შორის, უფრო სწრაფად ხდება მონაცემთა ბეკაპირება, უფრო სწრაფია ერთი სერვერიდან მეორე სერვერზე გადართვა, უფრო სწრაფია სერვერის გადატვირთვა და უფრო მარტივია ვირტუალურ მანქანების მართვა ფიზიკურ სერვერებთან განსხვავებით;

4. ზემოთ ჩამოთვლილი შედეგების მიხედვით და კონკრეტულად, ინფორმაციის სწრაფად გაცვლის საფუძველზე გაუმჯობესდა პროგრამული უზრუნველყოფის ხარისხი მისი ექსპლუატაციის ეფექტურობის თვალსაზრისით;

5. სხვა ოპერაციულ სისტემებთან შედარებით განსაკუთრებით ეფექტურია Linux ოპერაციული სისტემის გამოყენება, რადგან იგი

სტაბილურობის და სისწრაფის თვალსაზრისით ყველაზე საუკეთესო ოპერაციული სისტემაა ვირტუალიზაციის პირობებშიც;

6. დიდი განაწილებული სისტემების შემთხვევაში, როცა მონაცემთა ბაზები საკმაო მოცულობისაა (მაგ., 2-3 ტერაბაიტი ან მეტი), რეკომენდაცია შეიძლება გავუწიოთ Oracle-ს მონაცემთა ბაზების მართვის სისტემას, რომელიც დღესდღეობით ერთ-ერთი ყველაზე მძლავრი ქსელური მონაცემთა ბაზაა.

გამოქვეყნებული ლიტერატურა:

1. **ჩერქეზიშვილი გ.** ვირტუალიზაცია მონაცემთა დამუშავების ცენტრებში. სტუ-ს შრ.კრ., „მართვის ავტომატიზებული სისტემები“, N1 (21). თბ., **2016**. გვ.260-263.

2. სურგულაძე გ., ოდიშარია კ., ფხაკაძე ც., კეკენაძე ა., **ჩერქეზიშვილი გ.** საფინანსო ორგანიზაციის ბიზნესპროცესების და IT-სამსახურის ინფორმაციული უსაფრთხოების რისკების შეფასება. სტუ-ს შრ.კრ., „მართვის ავტომატიზებული სისტემები“, N1 (21). თბ., **2016**. გვ.248-253.

3. სურგულაძე გ., გულიტაშვილი მ. **ჩერქეზიშვილი გ.** პროგრამული უზრუნველყოფის ტესტირების ტიპები და სცენარები. სტუ-ს შრ.კრ., „მართვის ავტომატიზებული სისტემები“, N1 (14). **2013**. გვ.95-99.

4. სურგულაძე გ., **ჩერქეზიშვილი გ.** საპროცესინგო სისტემებში პრობლემების მართვის ავტომატიზაცია. სტუ-ს 90 წლისადმი მიძღვნილი საერთაშორისო სამეცნ. კონფერენცია შრ.კრ. „21-ე საუკუნის მეცნიერებისა და ტექნოლოგიების ძირითადი პარადიგმები“. სტუ, თბილისი, სექტ., **2012**. გვ.30-33.

5. ჩერქეზიშვილი გ. პროგრამების ხარისხის მართვა ვირტუალიზაციის პირობებში. სტუ-ს სტუდენტთა 84-ე ღია საერთაშორისო სამეცნიერო კონფერენცია.

6. შუბითიძე ა., შუბითიძე ნ., ბიტარიშვილი მ., **ჩერქეზიშვილი გ.** შრომითი დასაქმების სააგენტოს ბიზნეს-პროცესების მოდელირება UML/2 ტექნოლოგიით და მონაცემთა ბაზის დაპროექტება. **საერთაშ.სამეცნ.კონფ.**

„მართვის ავტომატიზებული სისტემების და ახალი ინფორმაციულ ტექნოლოგიები“. 20-22 მაისი. სტუ, თბილისი. **2011**. გვ. 113

7. გულიტაშვილი მ., **ჩერქეზიშვილი გ.** WEB აპლიკაციების დამუშავების პროცესის მოდელირება UML/2 ტექნოლოგიით. სტუ-ს შრ.კრ., „მას“ N1 (10). **2011**. გვ.180-184.

8. სურგულაძე გ., გულიტაშვილი მ., კაკულია ი., **ჩერქეზიშვილი გ.**, ჯავახიშვილი ი. პროგრამული სისტემების, სასიცოცხლო ციკლის მოდელირება უნივერსალური და ექსტრემალური პროგრამირების პრონციპების კომპრომისული გადაწყვეტით. სტუ-ს შრ.კრ., „მას“ N1 (8). **2010**. გვ.63-70.

ABSTRACT

In this thesis discuss the issues of software quality management under organizational management using computer systems based on virtualization. Also, analyzed and classified software quality management methods, evaluation criteria and mechanisms. Considered Of computer networks and topologies in data centers. Posed of models of quality management with ISO standards. Also, for determination of quality in this paper discuss the principals and advantages of virtual servers and virtual environment than physical servers. The table include the main parameters of quality

There is proposed to solve the problems with modern information technologies, in particular: The benefits of management of VMware, Oracle and Linux in virtual environment for distributed systems, while physical servers. With practical examples and observations there are some recommendations for solving the problems. The paper also provides management process of software systems life cycle, such as the system of business requirements definition, analysis of object-oriented and solution the object-oriented design tasks, the technical project tasks

preparation for developers, testing and implementation in the client organization. Each task is discussed as IT- service and it serve the fulfillment of the organization's business processes.

In that thesis also described the main parameter of software securing, to resolve the system requirements. Its business process, business rules and modals implemented based on unified modeling language UML. There were created UseCase and Activity Diagrams. Defined the roles of software project manager, business analysts, designers, developers, testing specialist and etc.

There also resolve the task of server virtualization, in cycle of software lifecycle modal for distributed systems, with components (Component-D) and deployment (Deployment-D).

Described suitable calculating of software quality for cases of traditional (physical severs) and virtualization.

Described the results of comparison in servers of software systems, to exchange information in time, financial cost, the time of system recovery and time of system backup.

In the dissertation represented the results of experiments to switch from physical servers to virtual servers. It's evaluated with level of quality of software systems in specific period of time in condition of virtual and physical servers. Corresponding the graphical diagrams. described also petri nets with its objects and diagrams.

Considered also Oracle database objects, what details where tested to switch on virtual server from physical, what was the main tasks to receive a stable and high-quality results.