LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

DALĪTĀS PARALĒLĀS BOJĀJUMPIECIETĪGĀS FAILU SISTĒMAS

BAKALAURA DARBS

Autors: **Eduards Kļaviņš**

Studenta apliecības Nr: EK11089

Darba vadītājs: Dr. Dat. Leo Trukšāns

RĪGA 2015

ANOTĀCIJA

Bakalaura darba tēma ir „dalītās paralēlās bojājumpiecietīgās failu sistēmas”. Darba mērķis ir iepazīties ar šāda veida failu sistēmām, izprast, kā tās darbojas un kādi ir to plusi un mīnusi. Darba gaitā ir uzstādīta viena no šādām failu sistēmām (Ceph) uz vairākiem serveriem, lai varētu izpētīt tās darbību un arhitektūru, kā arī izprast tās darbību.

Atslēgvārdi: Failu sistēmas, Ceph, OSD, CloudStack, Zabbix

ABSTRACT

Topic of this Bachelor thesis is “distributed parallel fault-tolerant file systems”. Main goal of this paper is to understand the use of these file systems and understand how they work and what are their main advantages and disadvantages. During the development of this paper, one of these file systems (Ceph) has been set up on multiple servers to make the exploration easier.

Keywords: File systems, Ceph, OSD, CloudStack, Zabbix

SATURA RĀDĪTĀJS

TERMINI UN SAĪSINĀJUMI

Replikācija – informācijas dalīšanās ar vairākiem resursiem (piemēram, cietie diski), lai uzlabotu uzticamību, bojājumpiecietību un pieejamību.

Migrācija – datu pārraidīšanas process starp dažādām glabātuvēm vai sistēmām.

Mērogojamība – sistēmas spēja pielāgoties pie lielākas slodzes, piemēram, pieliekot papildus mezglu pie kādas vairāku mezglu skaitļošanas sistēmas, sistēma jauno mezglu iekļauj un izmanto skaitļošanā un kļūst jaudīgāka.

Klasteris – savienoti datori, kas savā starpā strādā kopā. Varētu šķist, ka tā ir kā viena vesela sistēma, kaut gan sastāv no vairākiem atsevišķiem datoriem.

Ugunsmūris (*ang.v. firewall*) - ugunsmūris ir aparatūra vai programmatūra, kura pārbauda informāciju, kas ienāk no interneta vai tīkla, un atkarībā no ugunsmūra iestatījumiem to bloķē vai arī ļauj ienākt datorā.

SSH (*Secure Shell*) - tīkla protokols, kas ļauj izmainīt datus starp divām tīkla iekārtām, izmantojot aizsargātu kanālu.

OSD (*Object Storage Device*) – fiziskā vai loģiskā glabātuves vienība.

Ceph OSD – programmatūra, kas darbojas ar loģisko disku OSD.

Ceph monitori – programmatūra, kas pārrauga Ceph sistēmu un glabā klasteru karti.

Strīpošana (ang.v. *Data Striping*) – loģiski sekvenciālu datu segmentēšana tā, ka katrs segments tiek noglabāts uz citas fiziskas glabātuves.

Inode - Datu struktūra UNIX sistēmās, kurā glabājas failu un direktoriju pamatinformācija.

MDS (*Ceph Metadata server*) – programmatūra, kas uzglabā Ceph failu sistēmas metadatus (direktorijas, failu atļaujas un tamlīdzīgi).

Pūls (ang.v. *pool*) – loģiskās partīcijas objektu glabāšanai.

Izvietojuma grupa (PG) (*placement group*) – kad datus ievieto klasterī, objektus sagrupē pa izvietojuma grupām (PG), kuras piesaista OSD. Tas samazina objektu metadatu apjomu, kā arī palaistos procesus (būtu neefektīvi izsekot katra individuāla objekta novietojumam, tāpēc tie ir sagrupēti).

Kodols *(ang.v. Kernel)* - Linux sastāvdaļa, kas nodrošina sadarbību starp aparatūru un programmatūru, resursu sadali un procesu vadību.

RADOS (*Reliable, Autonomous, Distributed Object Store*) – Ceph glabātuves klasteris, kas ir objektu uzglabāšanas ierīce, kas var mērogoties līdz pat tūkstošiem ierīču, tā ir daļa no Ceph, kas sevī iekļauj visu serveru kopumu un kalpo kā datu glabātuve, kas sevi spēj uzraudzīt.

RAID (*redundant array of independent disks*) – datu uzglabāšanas virtualizācijas tehnoloģija, kas apvieno vairākus diskdziņus vienā loģiskā vienībā, lai paaugstinātu veiktspēju un iespējotu redundanci.

FastCGI – protokols, kas kalpo kā saskarne ar tīmekļa serveri.

HTTP (*Hypertext Transfer Protocol*) – protokols, ko lieto globālais tīmeklis savstarpējai komunikācijai.

API (*Application Programming Interface*) – likumu, nosacījumu kopa, kam programmatūras seko, lai varētu savstarpēji sazināties. Tā ir kā saskarne starp dažādām programmatūrām.

GB (gigabaits) – atmiņas ietilpības mērvienība, tā sastāv no 1024 megabaitiem, visu lielāko datu glabāšanas ierīču ietilpību mēra ar šo mērvienību.

GHz (gigahercs) – procesora takts frekvences mērvienība.

SATA – cietā diska veids, kurš ,darba ietvaros, griežas ar 7200 apgriezieniem minūtē.

SAS - cietā diska veids, kurš ,darba ietvaros, griežas ar 15000 apgriezieniem minūtē.

Žurnāls – failu sistēma, kas seko līdzi izmaiņām, kas vēl nav ierakstītas ilgtermiņa glabātuvē (šī darba ietvaros – OSD).

root – lietotājs unix sistēmās, kam ir pieeja visām komandām un failiem.

Repozitorijs - centrālā vieta, kurā organizēti tiek veidots un uzturēts datu sakopojums. Parasti šī vieta ir datora atmiņa. Repozitorijā var būt gan vieta, kas tieši pieejama lietotājiem, gan arī vieta, no kuras var iegūt specifiskas datu bāzes datnes vai dokumentus, lai tos pārvietotu vai izplatītu datoru tīklā.

Pakotne – datne, kuru izmanto pakotņu pārvaldnieks, lai uzstādītu kādu servisu vai bibliotēku.

VPN *(ang.v. Virtual private network)* – virtuālais privātais tīkls ir tehnoloģija kas nodrošina vienu vai vairākus tīkla savienojumus (virtuālu tīklu) kāda tīkla vai vairāku tīklu (piemēram, interneta) ietvaros.

IEVADS

Dalītās failu sistēmas, kas ir reizē gan paralēlas, gan bojājumpiecietīgas, strīpina un replicē datus pār vairākiem serveriem, lai sasniegtu augstu veiktspēju un saglabātu datu integritāti. Pat ja kāds no serveriem cieš neveiksmi, dati netiek pazaudēti. Šāda veida failu sistēmas izmanto gan augstas veiktspējas skaitļošanā, gan arī augstas pieejamības klasteros. Šādas failu sistēmas liek galveno uzsvaru uz augstu pieejamību, mērogojamību un augstu veiktspēju [1].

Šādām failu sistēmām var būt neskaitāmi daudz pielietojumu dēļ augstāk minētajām raksturīpašībām. Es strādāju uzņēmumā, kas nodarbojas ar serveru, mājas lapu uzturēšanu, serveru īri un administrēšanu, kā arī mākoņdatošanas pakalpojumiem [2]. Viens pielietojums, kas piesaistīja uzmanību, ir šādu failu sistēmu integrācija ar "CloudStack" virtualizācijas platformu, kuru lieto uzņēmums, kurā strādāju. Šāda veida integrācija atļauj izmantot šīs failu sistēmas kā virtuālo mašīnu glabātuvi. Tas būtu noderīgi, jo šāds risinājums nodrošinātu augstu veiktspēju ar bezgalīgu mērogojamību [3]. Vēl ir arī iespējams to izmantot kā vienkāršu failu sistēmu, kurā uzglabāt kritiski svarīgus datus, kuriem jābūt visu laiku pieejamiem, kas vienmēr var noderēt.

Darba mērķis ir izprast šādu sistēmu darbību, kā arī tās pielietojumu, vai tas būtu reāli ieviešams produkcijā. Darba gaitā plānots izvēlēties kādu konkrētu failu sistēmu, uzstādīt to testa vidē un pārbaudīt tās stabilitāti un ātrdarbību.

1. DARBA MĒRĶI UN UZDEVUMI

Darba mērķis ir izprast dalītu paralēlu bojājumpiecietīgu failu sistēmu darbību, kā arī tās pielietojumu, vai tas būtu reāli ieviešams produkcijā. Darāmos darbus var iedalīt 4 dažādos posmos:

- šo failu sistēmu izpēte un apgūšana, izprašana;
- darba vides sagatavošana;
- nepieciešamo servisu uzstādīšana un konfigurēšana;
- failu sistēmas testēšana.

Lai sasniegtu darba mērķi, vispirms tika izdalīti un apkopoti secīgi veicamie uzdevumi:

- Apgūt, izpētīt un izprast dalītas paralēlas bojājumpiecietīgas failu sistēmas;
- Apgūt CloudStack un tā lietotāja saskarni;
- Izveidot jaunus serverus ar Linux CentOS 7 operētājsistēmu;
- Piešķirt tiem publiskās IP adreses;
- Konfigurēt gan CloudStack, gan CentOS ugunsmūri;
- Pārbaudīt savienojamību ar internetu un savstarpēji ar serveriem;
- Uzstādīt un konfigurēt Ceph;
- Pieslēgt monitoringu;
- Testēt izveidoto klasteri.

Lai izpildītu šos uzdevumus, bija nepieciešams apgūt teorētiskās zināšanas sekojošās jomās:

- Kas ir CloudStack un kā tas darbojas;
- Kā rīkoties ar CloudStack, lai izveidotu jaunus serverus un pievienotu tiem tīklu;
- Linux operētājsistēma un Linux tehnoloģijas;
- Failu sistēmu veidi;
- Ceph arhitektūra;
- Kas ir bloku ierīce;

- Kas ir objektu glabātuve;
- Izmaiņas CentOS 7 versijā;
- Kas ir systemd un kā tas strādā, ar ko atšķiras no init.d;

1.1. Problēmas ietekmes analīze uz IS struktūru kopumā

Darba mērķis radies personīgas intereses nolūkos un tam ir potenciāls uzlabot esošo uzņēmuma infrastruktūru. Tiek izveidota failu sistēma testēšanas nolūkos, lai pārbaudītu, vai tai ir potenciāls, lai to pielietotu produkcijā. Uzlabojot uzņēmuma infrastruktūru, tas uzlabotu iespējas piesaistīt potenciālos klientus.

2. RISINĀJUMA PLĀNOŠANA

Šī nodaļa veltīta izvēlētā risinājuma plānošanas un izstrādes aprakstam pirms reālu darbību veikšanas. Plānošanu var sadalīt dažādos posmos – failu sistēmu izpēte un saprašana; darba vides sagatavošana, izveidojot CloudStack instances un visu nepieciešamo failu sistēmu un lietotņu uzstādīšana un konfigurēšana priekš šīs failu sistēmas, klastera testēšana.

Pirmkārt, nepieciešams izvēlēties kādu no šīm failu sistēmām, kuru uzstādīt un izmēģināt. Otrkārt, ir jāizpēta šī failu sistēma, jāsaprot, kā tā strādā. Failu sistēmas izveide turpinās ar jaunu serveru izveidošanu iekš CloudStack, to konfigurēšanu , izvēloties katra servera jaudu atbilstoši prasībām (procesoru skaits, operatīvās atmiņas apjoms, cietā diska izmērs, tīkls, operētājsistēma). Pēc tam nepieciešams uzstādīt visu, kas vajadzīgs, lai failu sistēma strādātu bez problēmām, tālāk atliek nodrošināt serveru drošību un pārbaudīt, kā šī failu sistēma strādā un vai tā varētu būt noderīga uzņēmuma attīstībai.

2.1. Izmantotie rīki un tehnoloģijas darba izpildē

- CloudStack
- iptables
- systemd
- Ceph
- ceph-deploy
- rbd
- Zabbix

3. FAILU SISTĒMAS IZVĒLE

Šajā nodaļā paredzēts apskatīt un izprast šīs failu sistēmas, kā arī izvēlēties vienu no tām, kuru varētu testēt. Zemāk ir aprakstīts, kādas atsevišķi ir dalītas, paralēlas kā arī bojājumpiecietīgas failu sistēmas. Šis darbs ir par failu sistēmām, kas apvieno visu šo 3 kategoriju īpašības un atbilst tām.

3.1. Dalītās failu sistēmas

Šādām failu sistēmām nav pieejas bloku līmenī vienai un tai pašai glabātuvei, tās lieto tīkla protokolu. Tās mēdz saukt arī par tīklu failu sistēmām, kaut gan tās nav vienīgās, kas izmanto tīklu, lai sūtītu datus. Dalītās failu sistēmas var liegt piekļuvi failu sistēmai ar pieejas sarakstiem vai atslēgām gan uz serveriem, gan klientu pusē, atkarībā no protokola.

Šo sistēmu galvenā īpašība ir tā, ka failiem var piekļūt no lokālā datora tieši tā pat kā mēs piekļūtu lokāliem failiem – iespējams šo failu sistēmu gan piemontēt, gan nomontēt, uzskaitīt direktorijas, rakstīt/lasīt baitu robežās, izmantot lokālās sistēmas atļaujas. Šādu failu sistēmu galvenais mērķis, uz ko visas šādas sistēmas tiecas, ir būt "neredzamām" vairākos aspektos [4]:

- Pieejas neredzamība – klientiem nav jāzina, ka faili ir dalīti pa tīklu; klienti tiem var piekļūt tāpat kā lokāliem failiem.
- Atrašanās vietas neredzamība – faila nosaukums „nenodod" tā atrašanās vietu, tie ir redzami un nosaukti tāpat kā lokālie faili.
- Laiksakritības neredzamība – visiem klientiem failu sistēma izskatās vienādi, t.i., ja kāds process veic izmaiņas kādā failā, visi citi procesi tai pašā sistēmā vai attālinātā sistēmā, kas piekļūst failam, redzēs izmaiņas momentāni.
- Atteices neredzamība – klientam un viņu programmām servera atteices gadījumā nevajadzētu redzēt izmaiņas.
- Neviendabīgums - failu servisam vajadzētu būt iespējai uzstādīt to uz dažādām operētājsistēmām un detaļām.
- Mērogojamība – failu sistēmai vajadzētu labi darboties gan mazos apmēros (sākot ar

pāris mašīnām) un mērogoties līdz milzīgām (vairāki tūkstoši).

- Replicēšanas neredzamība – lai atbalstītu mērogojamību, iespējams būtu nepieciešamība replicēt datus uz vairākiem serveriem, klientiem tas nebūtu "jājūt" vai jāzina.
- Migrācijas neredzamība – failus var pārvietot starp serveriem, klientiem tas nebūtu "jājūt" vai jāzina.

3.2. Paralēlās failu sistēmas

Šādas failu sistēmas ir parasti saslēgtas klasterī un datus glabā uz vairākiem glabātuves mezgliem, parasti priekš redundances un veiktspējas uzlabošanas [4]. Paralēlās failu sistēmas izdala viena objekta datus pār vairākiem mezgliem. Bieži arī šīm sistēmām glabātuve atrodas fiziski citā vietā prom no skaitļošanas sistēmas. Tās veidotas priekš augstas veiktspējas un lieto ātrgaitas tīklus datu pārraidei. Veidotas, lai būtu paralēlas – lai failam vienlaicīgi var piekļūt dažādi klienti [5].

3.3. Bojājumpiecietīgās failu sistēmas

Tās ir tādas failu sistēmas, kas ļauj sistēmai turpināt strādāt pat ja ir bijusi kāda kļūme vai kāds no mezgliem ir atteicies. Sistēmas ātrums var samazināties, taču pati sistēma neatteiksies pilnībā kā tas ir parastās sistēmās, kur jebkura mazākā kļūme var izraisīt pilnīgu sistēmas atteici [6]. Šādās sistēmās atkopšanās pēc kļūdām parasti iedalās 2 dažādos veidos kā tas notiek - *roll-forward* (sistēma "nopauzē", patērē kādu laiku lai sistēmu salabotu un turpina tālāk strādāt) un *roll-back* (atgriež sistēmu kādā iepriekšējā stāvoklī, kurā nebija kļūdu).

Veidojot failu sistēmu, ir jārēķinās, ka būs bojājumi, tas jāuzskata par normu nevis izņēmumu. Lai panāktu bojājumpiecietību, tiek lietota failu replikācija, kopējot datus uz dažādiem serveriem – kad klients pieprasa datus, viņš "neredzami" piekļūst vienai no datu kopijām. Ir jāņem vērā arī datu sinhronizācija starp šīm kopijām, kuras tiek kopētas uz dažādiem serveriem – kad datus pārraksta, tad visām kopijām ir jābūt ar aktuālo informāciju. Ir 3 veidi, kā to panākt:

- Sinhronā metode – jebkurš pieprasījums pēc mainītajiem datiem tiek bloķēts, kamēr visas kopijas nav atjaunotas. Tas nodrošina to, ka lietotāji piekļūst datu jaunākajai versijai, bet var aizkavēt pieprasījumu izpildi.
- Asinhronā metode – pieprasījumi pēc mainītajiem datiem ir atļauti pat ja to kopijas nav aktuālākās versijas. Tādējādi lietotāji var pēc iespējas ātrāk piekļūt datiem, bet tie var nebūt paši aktuālākie.
- Pus-asinhronā metode – pieprasījumi tiek bloķēti, kamēr dažas no kopijām (ne visas) tiek atjaunotas. Pieņemsim, ka ir 5 datu kopijas, pieprasīt šos datus varēs, kad 3 no tām būs atjaunotas. Tas ierobežo iespēju piekļūt veciem datiem, tai pašā laikā samazinot aizkavi piekļūšanai.

Jāņem vērā arī slodzes izlīdzināšanas mehānismi, kuri balansē slodzi kad tiek bojāts kāds no mezgliem, kā arī tad, kad tas atgriežas atpakaļ. Parasti failu sistēmas lieto sarakstu kurā tiek ievietoti dati, kas tiek pārvietoti mezglu bojāšanās vai atgriešanās gadījumos. Algoritms periodiski iet pāri šim sarakstam un veic noteiktās darbības. Piemēram, Ceph failu sistēma lieto funkciju, ko sauc par *CRUSH* (Controlled Replication Under Scalable Hashing), kura nejauši saglabā jaunos datus, pārvieto esošo datu apakškopas uz jaunajām glabātuvēm un kopīgi atjauno datus no tām glabātuvēm, kas ir bojātas [7].

3.4. Pieejamās failu sistēmas

Mūsdienās ir vairākas šāda veida failu sistēmas, kas ir vienlaicīgi bojājumpiecietīgas, paralēlas un dalītas. Populārākās no tām, kuras lieto produkcijā un bieži atjaunina, ko varētu minēt ir HDFS, Ceph, GlusterFS, MogileFS. Katrai no šīm sistēmām ir pavisam atšķirīga arhitektūra. Dziļāk apskatīsim Ceph, jo tā ir bezmaksas un atvērtā pirmkoda, to var izmantot un uzstādīt jebkurā testa vidē - ja vēlas, tad to var izdarīt pat savās mājās. Visvienkāršākajai konfigurācijai pietiek tikai ar 3 serveriem vai pat 3 virtuālajām mašīnām, kuras visas darbojas uz viena datora. Tā ir arī lieliska platforma, lai iegūtu zināšanas par to, kā strādā šādas mērogojamas sistēmas, kā arī kā notiek objektu glabāšana, pat ja produkcijas vide izmanto ko citu.

4. CEPH ARHITEKTŪRA

Ceph vienā sistēmā piedāvā objektu, bloku un failu glabātuvi. Tā ir programmatūra, nav nepieciešama Ceph ražota aparatūra. Tā ir uzticama, viegli pārvaldāma, plaši mērogojama un bezmaksas (LGPL licence). Produkta dizains vērsts uz to, lai sistēmā nebūtu neviena vājā punkta un lai to varētu mērogot līdz neierobežotam mezglu daudzumam. Ceph mezgls ir atsevišķs serveris ar uzstādītiem dēmoniem uz tā un ir daļa no Ceph glabātuves klastera jeb RADOS, kas ir visu mezglu kopums, kuri savstarpēji komunicē un replicē, un dinamiski izplata datus. RADOS sastāv no 2 veidu dēmoniem:

- Ceph monitori – glabā klastera karti (serveru savstarpējo savienojumu karte); ja sistēmai ir vairāki monitori, tad tos arī var saslēgt klasterī un nodrošināt augstu pieejamību viena monitora atteices gadījumā.
- Ceph OSD dēmoni – pārbauda gan savu, gan citu OSD stāvokli un ziņo par to Ceph monitoriem. Atbild par rakstīšanas un lasīšanas operācijām.

Kad RADOS saņem datus no klienta, tie tiek glabāti kā objekti. Katrs objekts atbilst failam failu sistēmā, kurš glabājas uz OSD. Datus, ko klients ievieto glabātuvē, strīpo blokos un piesaista objektiem. Objektus identificē ar inode numuru plus objekta numuru un izvieto pa glabātuvēm ar speciālu algoritmu, ko sauc par *CRUSH*. Kad klients pieprasa datus, *CRUSH* sazinās ar Ceph klasteri, kurš atgriež inode numuru un faila izmēru. Tad klients var aprēķināt, no cik objektiem sastāv fails, savienoties ar OSD klasteri, lai saņemtu datus ar *CRUSH* algoritma palīdzību.

4.1. Mērogojamība un augsta pieejamība

Tradicionālajās arhitektūrās klienti komunicē ar centralizētu komponenti (vārteju, API, u.tml.), kas kalpo kā vienīgais ieejas punkts, un sarežģītu apakšsistēmu. Tas ierobežo veiktspēju un mērogojamību, kā arī rada vājo punktu visā sistēmā – ja šī sistēmas daļa atsakās, tad reizē arī visa sistēma ir nepieejama. Ceph izslēdz tikai vienu vienīgu ieejas punktu – klients var sazināties ar katru Ceph OSD dēmonu pa tiešo. Tie veido objektu replikas uz citiem Ceph mezgliem, lai nodrošinātu datu drošību un augstu pieejamību. Ceph arī izmanto monitoru klasteri, lai nodrošinātu augstu pieejamību starp OSD mezgliem. Pirms

Ceph klienti var rakstīt vai lasīt datus, tiem jāsavienojas ar Ceph monitoru, lai iegūtu visaktuālāko klastera karti. Visā Ceph glabātuves klasterī var būt viens šāds monitors, bet tad tas kļūst par sistēmas vājo punktu – ja tas atsakās, tad klienti vairs nevar ne lasīt, ne rakstīt datus. Lai paaugstinātu uzticību un pakalpojums būtu bojājumpiecietīgs, Ceph atbalsta monitoru saslēgšanu klasterī. Lai monitori varētu sinhronizēties līdz vienlīdzīgam stāvoklim, vairāk kā 50% no klasterī saslēgtajiem jādarbojas, ieteicams tos veidot nepāra skaitļu daudzumā. Piemēram, ja ir 5 Ceph monitori, tad trim no tiem ir jādarbojas [10].

4.2. CRUSH algoritms

To lieto gan Ceph klienti, gan Ceph OSD dēmoni, lai efektīvi izskaitļotu informāciju par objekta atrašanās vietu. Salīdzinot ar vecākiem algoritmiem, *CRUSH* nodrošina labāku datu organizācijas mehānismu un ļauj stipri lielu mērogošanu izdalot darbus visiem klientiem un OSD dēmoniem klasterī. Failu izvietošanu nosaka pēc hierarhiskas klastera kartes, kas atspoguļo pieejamos glabātuves resursus un satur loģiskos elementus, no kā tas ir izveidots. Piemēram, liela failu sistēma varētu sastāvēt no vairākām rindām ar serveru skapjiem, kas pildīti ar disku plauktiem, kuri ir pilni ar glabātuvēm. Iespējams arī definēt „izvietošanas noteikumus", kuros, piemēram, norāda, ka 3 failu replikas katra tiktu izvietota citā fiziskā serveru skapī, lai tie nedalītu vienotu elektrības pieslēgumu, nodrošinot lielāku drošību [9].

4.3. Klastera karte

Ceph sistēma paļaujas uz to, ka Ceph klienti un Ceph OSD dēmoni pārzina klastera topoloģiju, kas sastāv no 5 kartēm, kuras visas kopā dēvē par klastera karti:

1. Monitoru karte – sastāv no klastera unikālā identifikatora, katra monitora pozīcijas, porta un adreses. Satur arī informāciju par to, kad karte veidota un kad bija veiktas pēdējās izmaiņas.

2. OSD karte – satur klastera unikālo identifikatoru, izveidošanas un pēdējo izmaiņu laikus, pūlu sarakstus, repliku izmērus, PG numurus, OSD sarakstu un to statusu.

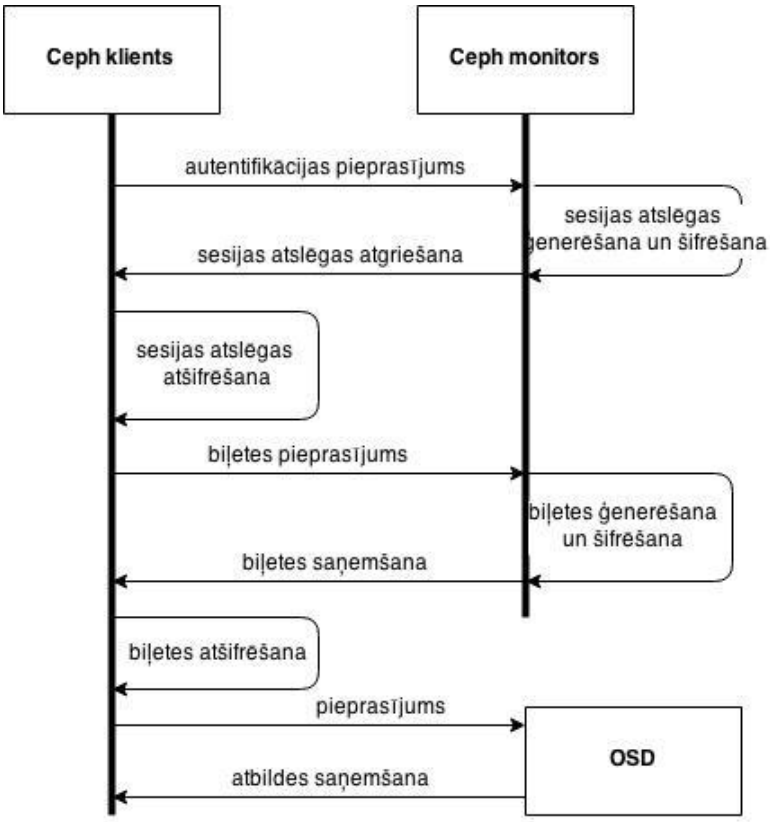
3. PG karte – satur PG versiju, izveidošanas un pēdējo izmaiņu laikus, katras izvietojumu grupas detaļas un stāvokli, un lietošanas statistiku.

4. *CRUSH* karte – satur glabātuves ierīču sarakstu, to hierarhiju un likumus, kā šo sarakstu šķērsot, kad jāuzglabā datus.

5. MDS karte – satur izveidošanas un pēdējo izmaiņu laikus, satur metadatu uzglabāšanas pūlu, metadatu serveru sarakstu un to stāvokļus.

4.4. Augstas pieejamības autentifikācija

Lai aizsargātu datus, Ceph nodrošina savu autentifikācijas protokolu *cephx,* kas autentificē lietotājus kas rīkojas ar Ceph klientiem. Tas notiek sekojoši – Ceph klients sazinās ar Ceph monitoru (katrs no monitoriem, ja tie saslēgti klasterī, var autentificēt lietotājus un izdalīt atslēgas, lai nebūtu viena vājā punkta), kas atgriež autentifikācijas datu struktūru, kura satur sesijas atslēgu, lai piekļūtu Ceph servisiem. To šifrē ar pastāvīgo slepeno atslēgu, lai tikai konkrētais lietotājs var veikt pieprasījumus monitoram. Klients tad izmanto šo atslēgu, lai veiktu pieprasījumus, monitors tam izsniedz biļeti, kas autentificē klientu ar OSD, kur

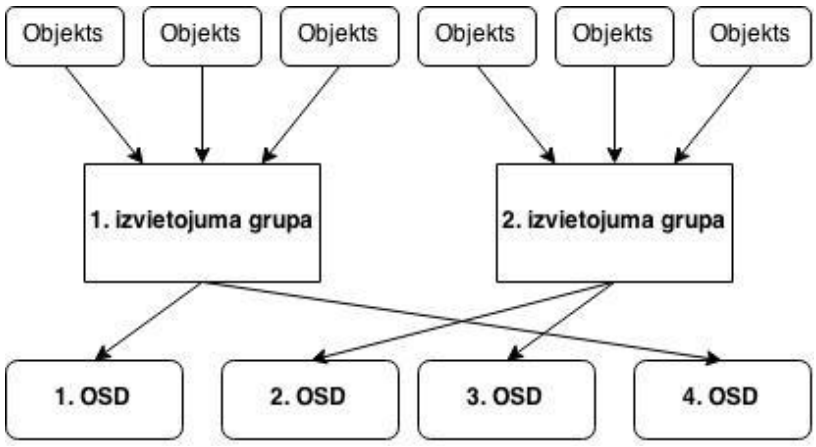


4.1. attēls. **Augstas pieejamības autentifikācija**

atrodas reāli dati. Tad Ceph klients atšifrē šo biļeti un izmanto to, lai piekļūtu gan OSD, gan MDS viscaur klasterī. Šis mehānisms uzskatāmāk redzams 4.1. attēlā zemāk.

4.5. Par pūliem un izvietojuma grupām

Ceph glabātuves sistēma atbalsta tādu jēdzienu kā pūli, kas ir loģiskās partīcijas objektu glabātuvei. Ceph klienti iegūst klasteru karti no monitoriem un raksta objektus pūlos. Pūla izmērs (repliku skaits), *CRUSH* likumi, izvietojuma grupu (PG) skaits nosaka kā Ceph izvietos datus. Katram pūlam ir vairākas izvietojuma grupas, kuras *CRUSH* piesaista pie OSD dinamiski. Kad Ceph klients uzglabā objektus, tad *CRUSH* piesaistīs katru objektu izvietojuma grupai. Ceph klasterim jāvar augt vai samazināties. Ja Ceph klients zinātu, kuram OSD dēmonam ir kurš objekts, tad būtu cieša sadarbība starp dēmonu un klientu, bet tā vietā *CRUSH* algoritms piesaista katru objektu izvietojuma grupai un to piesaista vienam vai vairākiem OSD dēmoniem Tas ļauj Ceph dinamiski balansēties brīdī, kad klasterī parādās jauns OSD dēmons ar OSD ierīcēm. 4.2. attēlā redzams kā CRUSH algoritms piesaista objektus novietojuma grupām, un kā tās piesaista pie OSD. Ar klastera kartes kopiju un *CRUSH* algoritma palīdzību.



4.2. attēls. **Objektu piesaiste grupām, kas piesaistītas OSD**

4.6. Objektu atrašanās vietas noteikšana

Kad Ceph klients sazinās ar monitoru, tas iegūst aktuālāko klastera karti, ar kuru zina visu par monitoriem, OSD un MDS klasterī. Bet tas nezina absolūti neko par objektu atrašanās vietu – tā tiek aprēķināta. Vienīgie nepieciešamie ieejas dati ir objekta ID un pūls. Vienkāršāk sakot, Ceph glabā datus pūlos (piemēram, ar nosaukumu "pirmais"). Kad klients

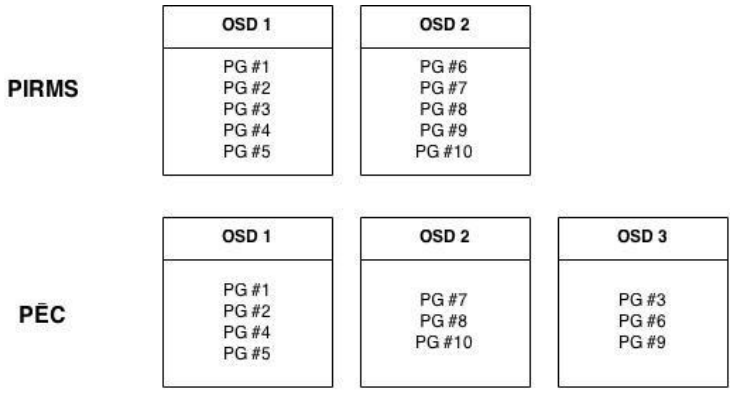
grib saglabāt objektu (piemēram, “Jānis”, “Pēteris” u.c.), tiek aprēķināta izvietojuma grupa, izmantojot objekta vārdu, jaucējkodu, izvietojuma grupu skaitu pūlā un pūla nosaukumu, kas pa soļiem būtu šādi:

1. Ceph klients padod pūla ID un objekta ID (*pūls=”pirmais”* un *objekta-id=”Jānis”*);
2. Ceph sajauc objekta ID ar jaucējkoda palīdzību;
3. Aprēķina izvietojuma grupas ID (piemēram, 42);
4. Ceph iegūst pūla ID no pūla nosaukuma (piemēram, “pirmais”=6);
5. Ceph saliek kopā pūla ID ar izvietojuma grupas ID (6.42).

Tādējādi Ceph iegūst objektu atrašanās vietas (tās aprēķina pats Ceph klients), kas ir daudz ātrāk kā veikt objektu atrašanās vietas vaicājumus. Ar CRUSH algoritma palīdzību, klients aprēķina objektu atrašanās vietu, kas tam ļauj pa taisno pieslēgties pie OSD, lai glabātu vai iegūtu objektus [11].

4.7. Vienmērīga sadalījuma atjaunošana

Kad tiek pievienots jauns Ceph OSD dēmons klasterim, tad tiek atjaunota klastera karte, jo tā mainās, līdz ar to mainās arī objektu izvietojums. 4.3. attēlā redzams vienmērīga



4.3. attēls. **Vienmērīga sadalījuma atjaunošana**

sadalījuma atjaunošanas process, kurā pie 2 OSD pieliek trešo (jo klasterī vairāk OSD mezglu, jo mazāk izmaiņu būtu, bet šai gadījumā izmaiņas ir ievērojamas) un dažas izvietojuma grupas migrē no esošajiem OSD uz jauno OSD 3. Lielākā daļa no izvietojuma grupām paliek sākotnējā atrašanās vietā un katrs OSD atbrīvojas no potenciālās slodzes.

4.8. Datu pastāvība

Daļa no tā, lai uzturētu datus pastāvīgus, ir objektu attīrīšana – Ceph OSD savstarpēji salīdzina objektu metadatus no vienas izvietojuma grupas ar tā replikām, kas glabājas izvietojuma grupās uz citiem OSD. Attīrīšana, kas parasti paredzēta vienu reizi dienā (var mainīt) atrod OSD kļūdas vai failu sistēmas kļūdas. OSD var arī veikt dziļāku attīrīšanu, salīdzinot datus objektos bitu pēc bita. Šo operāciju parasti veic reizi nedēļā un tā mēdz atrast bojātos sektorus, kuri netika atrasti pie vieglās attīrīšanas [12].

4.9. Ceph paplašināšana

Ceph var paplašināt jebkurš, kas to vēlas, atliek tikai izveidot objektu klases, kuras dēvē par Ceph klasēm. Kad pielieto klasi, var veidot jaunas objektu metodes, kuras izsauc jau iebūvētās Ceph objektu metodes vai citas metodes kas ir paša veidotas. Rakstīšanas operācijām Ceph klases var izsaukt iebūvētās vai paša veidotās metodes, veikt visdažādākās operācijas ienākošajiem datiem, visbeidzot ģenerējot rezultējošo rakstīšanas operāciju. Lasīšanas operācijām Ceph klases var izsaukt iebūvētās vai paša veidotās metodes, veikt visdažādākās operācijas ar izejošajiem datiem un atgriezt tos klientiem. Piemēram, var izveidot klasi, kas veic pārbaudi uz ienākošajiem datiem, atļauj rakstīt tikai nosacītus failu tipus (piemēram, tikai ar paplašinājumu "*.jpg*").

4.10. Ceph protokols

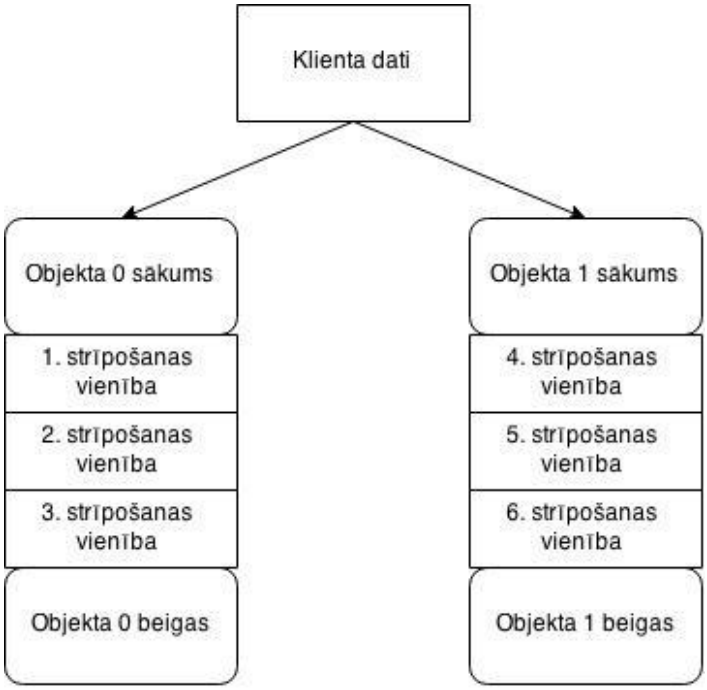
Ceph klienti izmanto iebūvēto protokolu, lai darbotos ar RADOS. Ceph šo funkcionalitāti iekļauj *librados* bibliotēkā, lai cilvēki varētu paši veidot savus Ceph klientus. RADOS nodrošina vienkāršu objektu glabātuves saskarni ar asinhronu komunikāciju. Šī saskarne nodrošina tiešu un paralēlu pieeju objektiem klasterī [13]:

- Darbības ar pūliem.
- Momentuzņēmumi un *Copy-on-write* (ja ir vairāki pieprasījumi vienam objektam, kas jau atvērts, citiem pieprasījumiem var izsniegt norādi uz kopiju šim failam. Ja kopija tiem mainīta, tad tai brīdī tiek arī izveidota patiesa kopija šim failam [14]) klonēšana.

- Objektu lasīšana/rakstīšana – izveidot vai dzēst (visu objektu) vai pievienot vai noņemt (baitu diapazonu).
- Izveidot, iestatīt, iegūt, aizvākt paplašinātos datnes atribūtus.
- Izveidot, iestatīt, iegūt, aizvākt atslēgu-vērtību pārus.
- Objektu klases.

4.11. Datu strīpošana

Datu uzglabāšanas ierīcēm ir caurlaidspējas ierobežojumi, kas ietekmē veiktspēju un mērogojamību, tādēļ glabātuves sistēmas bieži atbalsta datu strīpošanu, lai palielinātu caurlaidspēju un veiktspēju. Visbiežāk ar strīpošanu var sastapties iekš RAID. Ceph vislīdzīgākais RAID veids ir RAID 0. Ceph klients pārvērš datus no formāta, kādu redz klients uz objektiem priekš glabāšanas Ceph glabātuves klasterī. Ja ir vēlme veidot savu Ceph klientu, jāņem vērā, ka strīpošana ir jāievieš pašam. Ceph klienti raksta strīpošanas vienības



4.4. attēls. **Datu strīpošana objektos**

iekš glabātuves objekta līdz tas sasniedzis maksimālo kapacitāti, tad veido jaunu objektu priekš atlikušajām datu strīpām, kas attēlots 4.4. attēlā.

4.12. Ceph klienti

Ceph klients ir Ceph komponenšu kopums, kas var piekļūt Ceph glabātuves klasterim. Pastāv 3 dažādu veidu Ceph klienti – Bloku ierīce (RBD), objektu glabātuve (RGW), failu sistēma (Ceph FS). Katram no klientiem ir savs protokols, kas palīdz tiem sazināties ar Ceph glabātuves klastera protokolu *librados*, ko var redzēt 4.5. attēlā [15].

4.5. attēls. **Ceph augsta līmeņa arhitektūra**

4.12.1. Ceph bloku ierīce (RBD)

Viens bloks ir baitu sekvence, piemēram 512 baitu datu bloks. Bloku ierīce ir datoru glabātuves ierīce, kas atbalsta lasīšanu un rakstīšanu no fiksēta izmēra blokiem vai sektoriem. Tas ir visbiežāk sastopamais veids, kā glabā datus uz cietajiem diskiem, CD, disketēm, kā arī kasetēm [16]. Ceph bloku ierīce ir virtuāla ierīce, kas glabā savus datus blokos, strīpojot tos uz OSD. Tieši bloku ierīces ir laba izvēle virtualizācijai un mākoņdatošanai. Ceph bloku ierīce tiek veidota, lai uz tās glabātu, piemēram, virtuālās mašīnas QEMU vai KVM (virtualizācijas platformas) gadījumā (pagaidām citas nav atbalstītas, bet ir plānotas drīzumā). Ar RBD klientu, kurš izmanto *librbd* bibliotēku, piemēram, uz OSD var konfigurēt btrFS, ext4 vai XFS failu sistēmas, kuras var izmantot svarīgu datu glabāšanai [17].

4.12.2. Ceph objektu glabātuve (RGW)

Ceph Objektu glabātuves dēmons *radosgw* (kas lieto *libgrw* bibliotēku, kas redzama 4.5. attēlā) ir FastCGI serviss, kas nodrošina HTTP API, lai glabātu objektus un metadatus.

Tas ir kā slānis, kas ir virs Ceph glabātuves klastera, ar savu datu formātu, lietotāju datubāzi, autentifikāciju un pieejas kontroli. Lai glabātu vai lasītu datus, var izmantot OpenStack Swift (objektu glabātuves serviss) saderīgu API vai arī Amazon S3 (objektu glabātuves serviss) saderīgu API. Ceph objektu glabātuve ar terminu "objekts" apzīmē datus, ko tā glabā, S3 un Swift objekti nav tādi paši kā objekti, ko Ceph raksta savā glabātuvē [18].

4.12.3. Ceph Failu sistēma (CephFS)

Tā nodrošina mērogojamu failu sistēmu ar POSIX saderību kā servisu. Šīs failu sistēmas failus piesaista objektiem, ko Ceph glabā Ceph glabātuves klasterī. Ceph failu sistēmas serviss iekļauj Ceph metadatu serveri jeb MDS, kas ir izvietots Ceph glabātuves klasterī. MDS jēga ir glabāt visus failu sistēmas metadatus iekš augstas pieejamības Ceph metadatu serveriem. MDS serveri noņem slodzi un pieprasījumu skaitu no OSD dēmoniem - lai veiktu tādas vienkāršas operācijas kā direktoriju mainīšana (*cd* komanda *unix* vidē) vai to uzskaitījums (*ls* komanda *unix* vidē), uz to atbildēs metadatu serveris. Ceph failu sistēma atdala datus no metadatiem, glabājot visus metadatus uz MDS, savukārt visus datus vienā vai vairākos objektus Ceph glabātuves klasterī. Iespējams arī konfigurēt MDS serverus klasterī, nodrošinot augstu pieejamību, kur papildus mezgli var būt gaidstāves režīmā gatavi pārņemt vadību ja kas notiek ar aktīvo mezglu, var arī iestatīt, ka visi mezgli būs aktīvi, lai tie savstarpēji balansētu slodzi [19].

5. SERVERU IZVEIDE

Šai nodaļā tiks aprakstīts serveru izveidošanas process, uz kuriem tiks uzstādīta Ceph programmatūra. Tie būs virtuāli serveri, kas izveidoti ar CloudStack platformas palīdzību. CloudStack ir uzlikts virs XenServer hipervīzora. Mērķis ir izveidot nelielu klasteri failu uzglabāšanai. Pēc tam pārbaudīt, kas notiek, kad kāds no mezgliem iet bojā, vai sistēma atsāk darbību. Lai klasteris būtu bojājumpiecietīgs, ir izvēlēts risinājums ar 3 monitora serveriem un 3 OSD serveriem, kā arī 1 serveri, kas paredzēts Ceph administrēšanai.

5.1. Apache CloudStack

Tā ir atvērtā pirmkoda programmatūra, ko veidojis „Apache”. Tā ir paredzēta, lai izveidotu un pārraudzītu dažādu izmēru virtuālo mašīnu tīklus, kā arī tā ir augstas pieejamības, mērogojama IaaS (Infrastructure as a Service) jeb infrastruktūras pakalpojumu mākoņdatošanas platforma. To lieto gan lielie pakalpojumu sniedzēji, gan arī kompānijas priekš savām iekšējām darbībām. CloudStack sevī ietver visu, kas nepieciešams no IaaS mākoņa:

- Strādā ar serveriem, uz kuriem uzstādīti XenServer/XCP, KVM, Hyper-V, Vmware hipervīzori;
- Draudzīga lietotāju saskarne, kurai piekļūst no interneta pārlūkprogrammas;
- Nodrošina API;
- Pārvalda izveidoto serveru glabātuves, kā arī šablonus, ISO attēlus un momentuzņēmumus;
- Pārvalda tīklu servisus no datu posma (2.) slāņa līdz pat programmas (7.) slānim, kas iekļauj tādus servisus kā VPN, ugunsmūri, DHCP, NAT un tamlīdzīgi;
- Izlietoto tīklu, glabātuves, skaitļošanas resursu uzskaite;
- Lietotāju pārvaldība un pieejas tiesības;

Arī mana uzņēmuma infrastruktūrā ir ieviests šāds pakalpojums, kas ir uzstādīts uz servera ar XenServer hipervīzoru. Bakalaura darbam tika izveidots atsevišķs konts, kas ir atdalīts no klientu kontiem, kurā es varu veidot savus serverus un tīklus darba vajadzībām.

Resursu ierobežojumi tika noņemti, lai var veidot neierobežoti daudz serveru un tīklu.

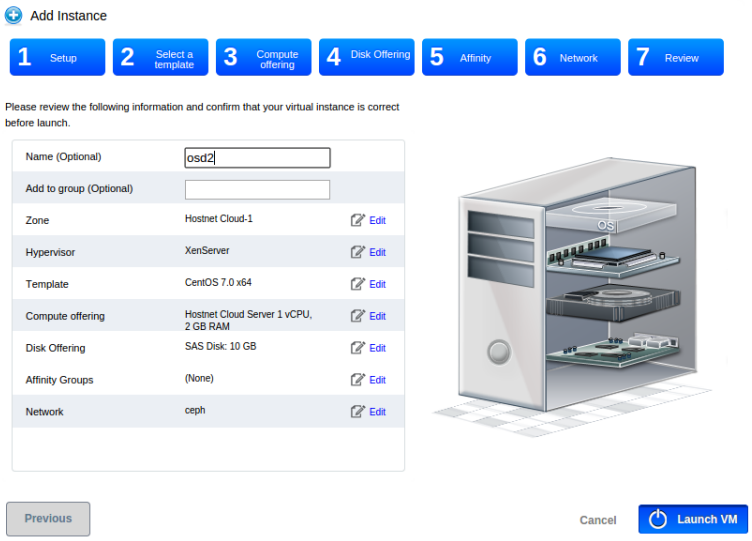
5.2. Operētājsistēma

Uz visiem serveriem tiks uzstādīta CentOS 7.0 (ang.v. Community ENTerprise Operating System) operētājsistēma, kas ir Linux distributīvs, dibināts kompānijā Red Hat, uz komerciāla RHEL (Red Hat Enterprise Linux). RHEL sastāv no bezmaksas programmnodrošinājuma ar atvērtu pirmkodu.

Parasti, lai uzstādītu jaunāko CentOS distribūciju, bija nepieciešama sistēmas pārinstalēšana, taču, sākot ar CentOS 7, tas vairs nav nepieciešams, operētājsistēmu varēs atjaunot uz jaunāko distribūciju. Šai versijā ir arī iekļauts konteineru atbalsts – var palaist vairākas Linux sistēmas uz viena resursdatora, bet, atšķirībā no virtualizācijas, katram konteinerim nav nepieciešama sava operētājsistēma, tie visi lieto vienu kodolu, sevī iekļauj tikai komponentes, kas nepieciešamas servisu vai programmatūru darbībai. XFS ir kļuvusi kā noklusētā failu sistēma, kas ļauj sistēmām mērogoties līdz pat 500 TB, kā arī paaugstina veiktspēju, īpaši paralēlām ievades, izvades operācijām. Arī *init* sistēma (servisu pārvaldnieks) tikusi nomainīta uz *systemd*, kura ir stipri pārāka vairākos rādītājos, taču pie tās ir jāpierod un tā ir arī jāizprot, lai saprastu, kā ar to ir jāstrādā [28].

Serverī izmantotā arhitektūra ir x86_64. Serveru sagatavošana ar Linux CentOS operētājsistēmu ir ļoti vienkārša, kura jāveic pa soļiem iekš CloudStack vadības paneļa:

- Pieslēdzamies CloudStack vadības panelim, ievadot lietotājvārdu, paroli un domēnu.
- Dodamies uz sadaļu “Instances” un pievienojam jaunu ar pogu “Add Instance”.
- Tālāk ir jāizvēlas, vai operētājsistēma tiks uzstādīta no ISO vai arī no šablona, manā gadījumā jāizvēlas šablons, jāpāriet uz nākamo soli.
- Tagad var izvēlēties, kuru no šabloniem lietot – es izvēlos Cenos 7.0 x64.
- Izvēlamies servera resursus – RAM un HDD apjomu.
- Izvēlamies papildus diska apjomu, nākamajā solī tīklu, kurā serveris būs pievienots.
- Nosaucam šo serveri un apstiprinām visu, lai serveris sāktu veidoties. Kopsavilkums vienam no serveriem (OSD 2) redzams 5.1. attēlā.



5.1. attēls. **Servera izveidošana iekš CloudStack**

5.3. Tīkla pievienošana

Lai serveriem varētu piekļūt ar SSH no VPN, nepieciešams pievienot katram no šiem serveriem publisko IP adresi:

- Jāpievieno 2 jauni tīkli – iekšējais un ārējais, jānorāda vārds, domēns un konts, kuram tos pievienot.
- Jāatver vaļā ārējā tīkla sadaļa un jāatver opcija, kur redzamas visas IP adreses (šobrīd tukšs).
- Jāiegūst 7 jaunas IP adreses – viena katram serverim.
- Jāatver IP adreses sadaļa un jāpiešķir tā kādam no serveriem.

Source CIDR	Protocol	Start Port	End Port	ICMP Type	ICMP Code	Add rule	State
	ICMP					Add	
89.111.13.198/32	ICMP			-1	-1		Active
89.111.13.198/32	TCP	22	22				Active

5.2. attēls. **CloudStack ugunsmūra konfigurācija**

- Jākonfigurē CloudStack ugunsmūris, kurā atvēru tikai 22. portu priekš SSH no savas

VPN IP adreses, kā arī pievienoju likumu, lai varu serveri pārbaudīt ar ping komandu, kas redzams 5.2. attēlā. Uz pašiem serveriem ir iptables ugunsmūris, uz kura tiks veikta papildus konfigurācija. Šai ugunsmūrī tiek nodrošināts, lai serveriem nevarētu piekļūt no citām IP adresēm ārpasaulē.

Serveriem piešķirtās iekšējās, ārējās un publiskās IP adreses, maskas un vārtejas, kā arī resursdatoru nosaukumi, ir redzami 5.1. tabulā.

Nosaukums	**Ārējā tīkla IP adrese, maska, vārteja**	**Iekšējā tīkla IP adrese, maska, vārteja**	**Publiskā IP adrese, maska, vārteja**	**Resursdatora nosaukums**
OSD 1	10.1.1.196, 255.255.255.0, 10.1.1.1	10.1.2.87, 255.255.255.0, 10.1.2.1	89.111.34.81, 255.255.255.0, 89.111.34.1	osd1.hostnet.lv
OSD 2	10.1.1.25, 255.255.255.0, 10.1.1.1	10.1.2.140, 255.255.255.0, 10.1.2.1	89.111.34.82, 255.255.255.0, 89.111.34.1	osd2.hostnet.lv
OSD 3	10.1.1.128, 255.255.255.0, 10.1.1.1	10.1.2.133, 255.255.255.0, 10.1.2.1	89.111.34.95, 255.255.255.0, 89.111.34.1	osd3.hostnet.lv
MON 1	10.1.1.86, 255.255.255.0, 10.1.1.1		89.111.34.102, 255.255.255.0, 89.111.34.1	mon1.hostnet.lv
MON 2	10.1.1.247, 255.255.255.0, 10.1.1.1		89.111.34.103, 255.255.255.0, 89.111.34.1	mon2.hostnet.lv
MON 3	10.1.1.75, 255.255.255.0, 10.1.1.1		89.111.34.107, 255.255.255.0, 89.111.34.1	mon3.hostnet.lv
Admin	10.1.1.165, 255.255.255.0, 10.1.1.1		89.111.34.98, 255.255.255.0, 89.111.34.1	manage.hostnet.lv

5.1. tabula. **Serveru nosaukumi un IP adreses**

5.4. Ugunsmūra konfigurācija

Šai gadījumā mums nebūs nepieciešams dinamiskais ugunsmūris firewalld, tāpēc to atslēdzam ar komandām `systemctl disable firewalld un systemctl stop firewalld.` Uzstādam iptables ugunsmūri ar komandu `yum install iptables-`

`services`, ieslēdzam to, lai tas palaistos pie servera ieslēgšanas ar komandu `systemctl enable iptables`. Tad nepieciešams atvērt tikai SSH savienojuma portus, kā arī portus, kas paredzēti priekš Ceph [29]. Uz OSD mezgla ir jāpievieno šādi iptables likumi:

```
iptables -A INPUT -i eth0 -m multiport -p tcp -s 10.1.1.0/24 --
dports 6800:7300 -j ACCEPT
```

```
iptables -A INPUT -i eth1 -m multiport -p tcp -s 10.1.2.0/24 --
dports 6800:7300 -j ACCEPT
```

Tie atver portus 6800-7300 priekš savstarpējās OSD komunikācijas gan iekšējā, gan ārējā tīklā. Uz monitora mezgliem papildus jāpieliek šis likums:

```
iptables -A INPUT -i eth0 -p tcp -s 10.1.1.0/24 --dport 6789 -j
ACCEPT
```

Tas atver 6789.portu ārējā tīklā, pa kuru notiek savstarpējā monitoru komunikācija.

6. SERVERU SPECIFIKĀCIJA

Linux ir spēcīga operētājsistēma, taču tai nepieciešami atbilstoši resursi, lai tā kvalitatīvi darbotos. Ar Ceph ir līdzīgi, tādēļ ir svarīgi izvēlēties pareizos resursus, lai serveri strādātu atbilstoši. Ceph tika veidots, lai tas varētu darboties uz pieejamas aparatūras, kas padara milzīgu klasteru būvēšanu un uzturēšanu ekonomiski izdevīgāku. Plānojot serveru specifikāciju, jāņem vērā dažādi faktori, kā ,piemēram, potenciālā veiktspēja. Ieteikums ir darbināt Ceph dēmonus uz serveriem, kas tam speciāli paredzēti, uz kuriem nav citas programmatūras, piemēram tīmekļa serveris vai e-pastu serviss [20].

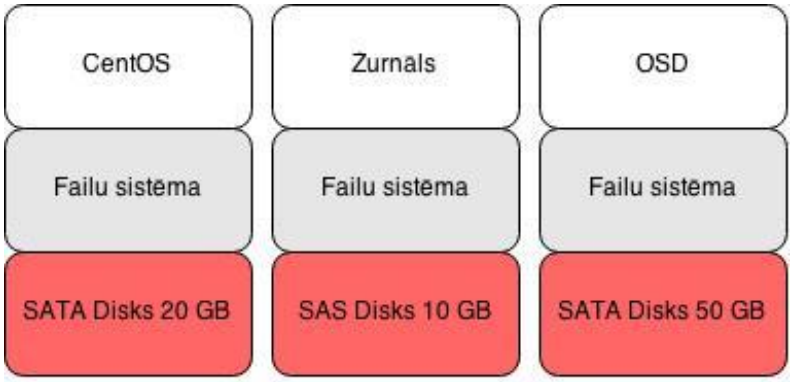
	OSD serveri	**MON serveri**	**Admin serveris**
Brīvpiekļuves atmiņa	2 GB	2 GB	1 GB
Procesors	2x Intel Xeon E5645 @2.40GHZ	Intel Xeon E5645 @2.40GHz	Intel Xeon E5645 @2.40GHz
SWAP atmiņa	2 GB	2 GB	1 GB
Operētājsistēmas disks	20 GB SATA	20 GB SATA	20 GB SATA
Journal disks	10 GB SAS	-	-
OSD disks	50 GB SATA	-	-

6.1. tabula. **Serveru specifikācija**

6.1. Cietais disks

Katram OSD serverim ir jābūt 1 cietajam diskam, SSD vai RAID grupai, uz kā glabāsies dati. Produkcijā OSD diskiem minimālais apjoms būtu 1TB, lai tie būtu finansiāli izdevīgi, jo lielāks cietais disks pēc izmēra, jo izdevīgāks tas sanāks rēķinot cenu par gigabaitu. Taču testam, lai taupītu vietu, tik lielus cietos diskus neveidosim, 6.1. tabulā un 6.1. attēlā redzami to izmēri. Slikta ideja ir arī uzstādīt vairākus OSD uz viena diska, arī ja tas sadalīts partīcijās, jo, ja fiziskajam diskam būs problēmas, tās ietekmēs reizē vairākus OSD. Tas pats attiecas uz monitoru un MDS mezgliem. Plānojot sistēmu, jāskatās uz disku pieejas, meklēšanas, lasīšanas, rakstīšanas ātrumiem kā arī uz kopējo caurlaidspēju. Šie fiziskie ierobežojumi var ietekmēt sistēmas veiktspēju, it īpaši pie disku atjaunošanās, kad tiem ir liela slodze. Ir ieteicams veidot vienu disku priekš operētājsistēmas un programmatūrām, kas

jāuzstāda, vienu disku priekš katra OSD dēmona, kurš būs uz servera. Lielākoties, lēnu OSD darbību izraisa tas, ka visi servisi izmanto vienu disku, kurš ir pārāk noslogots, tāpēc jau izdevīgāk ir pie plānošanas tos pareizi izdalīt. Manā gadījumā OSD serveriem būs 6.1. attēlā redzamais dizains ar 3 cietajiem diskiem. Operētājsistēmas diskam būs ext4 failu sistēma, jo tāda bija CloudStack šablonam, taču žurnāla un OSD diskiem būs XFS failu sistēma. Parasti Ceph uzstāda vai nu XFS vai BTRFS failu sistēmu, no šīm abām BTRFS ir daudz attīstītāka, taču tā vēl nav stabila, taču XFS ir pilnībā stabila un CentOS 7 atbalstīta, jo tā jau ir pieejama vairāk kā 20 gadus. Svarīgi ir arī tas, ka Ceph paši iesaka izmantot XFS produkcijā [23].



6.1. attēls. **Disku dizains OSD serveros**

Svarīgi ir arī izdalīt disku priekš žurnāla datiem, to izmanto diviem iemesliem – ātrumam un konsistencei. Ar žurnāla palīdzību, OSD dēmons mazas rakstīšanas operācijas var veikt ātrāk, jo tas šīs operācijas no sākuma ieraksta žurnālā, kura disks ir ātrāks, pēc tam lēnām pārliekot šos datus uz OSD diskiem. Runājot par konsistenci, Ceph OSD dēmons ieraksta žurnālā aprakstu par veicamo operāciju un tad to veic iekš failu sistēmas. Ik pēc pāris sekundēm OSD dēmons apstādina rakstīšanu un sinhronizē žurnālu ar failu sistēmu, dzēšot no žurnāla ierakstus par izpildītajām operācijām, ietaupot vietu. Kļūdas gadījumā, Ceph OSD dēmoni pārlasa žurnālu vēlreiz, sākot ar ierakstiem pēc pēdējās sinhronizācijas [21].

Bez šo disku optimizācijas un atsevišķas iedalīšanas, Ceph glabā žurnālu uz tā paša diska, kur OSD datus. Tas var izraisīt veiktspējas kritumu, jo , kad rakstīšanas operāciju jāpārliek no žurnāla uz OSD, tad noslogo vienu un to pašu disku, lai lasītu no žurnāla un rakstītu iekš OSD, dubultojot slodzi [22].

6.2. Procesors un operatīvā atmiņa

Ceph OSD serveri darbina RADOS servisu, kā arī aprēķina datu novietojumu ar CRUSH algoritmu, replicē datus un uzglabā klasteru karti, tādēļ OSD serveriem vajadzētu būt pietiekoši jaudīgiem (divkodolu procesori). Monitori tikai uzglabā klasteru karti, tādēļ tie nepatērē daudz procesora resursu. Jāņem arī vērā, ka uz servera darbosies CentOS operētājsistēma, kuras ātrai darbībai nepieciešami resursi, tādēļ OSD mezgliem ir izvēlēti 2 procesori, savukārt monitora mezgliem katram pa vienam, kas redzams 6.1. tabulā. Arī administrēšanas serverim ir tikai viens procesors, jo tas ir paredzēts tikai klastera administrēšanai, tātad, nelielai slodzei.

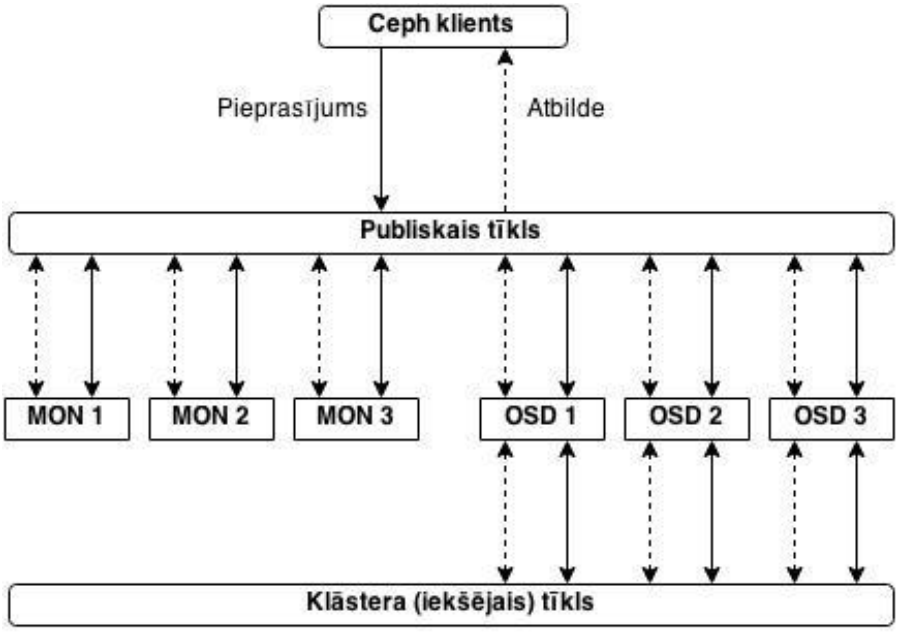
Monitoru serveriem jāspēj datus pasniegt ātri, tātad nepieciešams pietiekoši daudz operatīvās atmiņas (1 GB katram dēmonam), tādēļ, ņemot vērā, ka nepieciešama arī operētājsistēma, monitoru serveriem atvēlēts 2 GB operatīvās atmiņas. OSD serveriem nav nepieciešams tik daudz operatīvās atmiņas ikdienas operācijām, aptuveni 500 MB katram dēmonam, taču pie atjaunošanās, nepieciešams vairāk – aptuveni 1 GB katram glabātuves terabaitam. Kopumā, jo vairāk operatīvās atmiņas, jo labāk. Ņemot vērā CloudStack piedāvātos resursus, izvēlējos 2 GB RAM atmiņas.

6.3. Tīkls

Parasti šādi Ceph klasteri stipri noslogo tīklu. Katrā operācijā, īpaši rakstīšanā, ir iesaistīti vairāki mezgli. Kad datus glabā klasterī, vairāki bloki no vairākiem mezgliem tiek iesaistīti, kā arī tiem visu laiku jābalansē saturs un jāveic bloku replikācija. Produkcijas vidē visbiežāk tiek novēroti 10 gigabitu pieslēgumi, taču testa vidē izmantošu 1 gigabita tīklu, jo nav plānots stipri noslogot tīklu, kā arī 10 gigabitu tīkls nav pieejams.

Katram mezglam būs 3 pieslēgumi – viens būs paredzēts (publiskais), lai es varētu pieslēgties katram mezglam ar SSH attālināti, otrs, ārējais tīkls, pa kuru Ceph sazināsies. Trešais, iekšējais tīkls, ir OSD savstarpējai datu apmaiņai. Tas ir paredzēts veiktspējas uzlabošanai – OSD dēmoni veicu datu replikāciju, atjaunošanos un citas operācijas, kas var stipri noslogot tīklu.

Tas var radīt latentumu un veiktspējas kritumu, tādēļ labāk šīs operācijas veikt klastera (iekšējā) tīklā, kā redzams 6.2. attēlā.



6.2. attēls. **Ceph tīkli**

Tas uzlabo drošību – samazina ļaunprātīgu uzbrukumu iespēju [24].

7. CEPH UZSTĀDĪŠANA

Kad serveri ir izveidoti un tiem var piekļūt ar SSH palīdzību, tad var sākt Ceph uzstādīšanu uz šiem serveriem. Pirmkārt, nepieciešams uzstādīt serveriem jaunākos atjauninājumus ar komandu `yum update`. Kad visi atjauninājumi uzstādīti, tad uz visiem 6 Ceph mezgliem nepieciešams pievienot lietotāju ar root privilēģijām, ar ko notiks Ceph uzstādīšana:

```
useradd -d /home/ceph -m ceph
passwd ceph
echo "ceph ALL = (root) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/ceph
chmod 0440 /etc/sudoers.d/ceph
```

Kad tas ir izdarīts, tad ļoti svarīgi ir pareizi konfigurēt laika sinhronizēšanu, lai uz visiem serveriem būtu vienāds laiks, maksimālā atļautā laika nobīde Ceph sistēmā ir 0.05 sekundes. To dara uzstādot NTP lietotni (`yum install -y ntp ntpdate ntp-doc`), kas sinhronizē laiku ar speciāliem serveriem. Nepieciešams arī uzstādīt, lai lietotne tiktu palaista pie servera palaišanas. Lai nesarežģītu darbu, pēc Ceph dokumentācijas ieteikumiem, tiek izslēgts arī SElinux, ko var izdarīt iekš `/etc/sysconfig/selinux` faila.

Tālākas darbības tiks veiktas uz administrēšanas servera. Pirmkārt, nepieciešams konfigurēt SSH pieeju bez paroles izveidotajam ceph lietotājam [23]:

- Uz administrēšanas mezgla, jāizveido sava SSH atslēga ar tukšu paroli (komanda `ssh-keygen`).
- Jāpārkopē SSH atslēga uz katru mezglu ar komandu `ssh-copy-id ceph@mon2` (MON 2 mezgla gadījumā).
- Jāizveido `/home/ceph/.ssh/config` fails (ar 440 permisijām), lai administrēšanas mezgls var pieslēgties pārējiem kā ceph lietotājs ar šādu saturu (atkārtot visiem 6 mezgliem):

```
Host osd1
Hostname osd1
User ceph
```

Nākamajā solī nepieciešams sagatavot OSD diska partīcijas un žurnāla disku. Ar parted rīku tiek izveidotas GPT Patrīciju tabulas abiem diskiem. Ar `mkfs.xfs` komandu

nepieciešams uz visiem 3 OSD serveriem noformatēt OSD disku, lai tam būtu XFS formāts. Žurnāla diskam tiks izmantots neformatēts disks.

7.1. Ceph-deploy

Administrēšanas mezgls var pieslēgties visiem pārējiem Ceph mezgliem ar root privilēģijām. No šī mezgla tiek palaistas komandas, kas attiecas uz visu klasteri. Šāds mezgls nav obligāts, var arī savu personīgo datoru vai kādu no OSD, vai MON mezgliem izmantot administrēšanai. Lielākoties, uz šī mezgla tiks palaists ceph-deploy – rīks, kas speciāli veidots darbam ar Ceph klasteriem. Tas nav vienīgais veids, kā uzstādīt un izveidot klasteri, bet visparocīgākais.

Kad pieeja bez paroles no administrēšanas mezgla nodrošināta, nepieciešams arī pārliecināties, ka šis mezgls var sasniegt pārējos, piemēram, ar ping komandu. Kad savienojamība pārbaudīta, tad nepieciešams uzstādīt Ceph repozitoriju, kur iekš faila `/etc/yum.repos.d/ceph.repo` jāievieto sekojošs saturs, kur `{ceph-release}` ir Ceph versijas nosaukums, savukārt `{distro}` ir operētājsistēmas distribūcija, kas mūsu gadījumā attiecīgi būs hammer un el7 (apzīmē CentOS 7) :

```
[ceph-noarch]
name=Ceph noarch packages
baseurl=http://ceph.com/rpm-{ceph-release}/{distro}/noarch
enabled=1
gpgcheck=1
type=rpm-md
gpgkey=https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.
asc
```

Kad jaunais repozitorijs pievienots, tad nepieciešams uzstādīt rīka pakotni ar komandu `yum install ceph-deploy` un izveidot direktoriju, kur Ceph glabās savus konfigurācijas failus.

7.1.1. Klastera uzstādīšana

Pirmkārt, nepieciešams uzstādīt monitora mezglus ar komandu `ceph-deploy new mon1 mon2 mon3`. Pēc šīs komandas, tiek izveidots `ceph.conf` konfigurācijas fails. To

nepieciešams atvērt un pielikt klāt papildus konfigurāciju, kas redzama 1. pielikumā. Nepieciešams norādīt kā publiskā tīkla, tā klastera (iekšējā) tīkla adreses, kā arī cik bieži objektus replicēs un tamlīdzīgi. Kad konfigurācija ir sagatavota, tad var uzstādīt Ceph visos klasteros ar vienu komandu, kas tiek palaista no administrēšanas mezgla `- ceph-deploy install manage mon1 mon2 mon3 osd1 osd2 osd3`, kas izvadīs Ceph versiju pēc katra mezgla uzstādīšanas, norādot ka uzstādīšana bijusi veiksmīga. Pēc tam nepieciešams izveidots visus monitora mezglus un iegūt atslēgas: `ceph-deploy mon create-initial`.

7.1.2. OSD sagatavošana

Pirmkārt, OSD diskiem nepieciešams palaist `zap` komandu, kas izdzēš visu, kas tur varētu vēl būt: `ceph-deploy disk zap osd1:xvdc`. Kad tas darīts, tad izveido OSD uz diskiem, aiz kola norādot žurnāla disku`: ceph-deploy osd create osd1:xvdc:/dev/xvdb1` . Šīs abas komandas ir jāveic no administrēšanas servera un jāveic visiem 3 OSD serveriem. Visbeidzot, nepieciešams pārkopēt visas atslēgas un konfigurācijas failus no administrēšanas servera uz visiem pārējiem: `ceph-deploy admin manage mon1 mon2 mon3 osd1 osd2 osd3`.

Tagad varam pārliecināties, ka klasteri izdevies izveidot ar `ceph health` un `ceph status` komandām, lai aplūkotu to statusu. Komandas izvads redzams 7.1. attēlā. Ir redzami

```
[root@osd1 ~]# ceph health
HEALTH_OK
[root@osd1 ~]# ceph status
    cluster 00d51eb2-c068-45d7-a098-c960ff7abd2e
     health HEALTH_OK
     monmap e1: 3 mons at {mon1=10.1.1.86:6789/0,mon2=10.1.1.247:6789/0,mon3=10.1.1.75:6789/0}
            election epoch 8, quorum 0,1,2 mon3,mon1,mon2
     osdmap e64: 3 osds: 3 up, 3 in
      pgmap v136: 64 pgs, 1 pools, 0 bytes data, 0 objects
            104 MB used, 149 GB / 149 GB avail
                  64 active+clean
```

7.1. attēls. **Komandas "ceph status" izvads**

3 monitora serveri sadaļā `monmap` (monitoru karte), 3 OSD serveri un to stāvokļi sadaļā `osdmap` (OSD karte), kā arī sadaļā `pgmap` redzams cik izvietojuma grupu un pūlu ir uz servera.

7.2. Ceph bloku ierīces pievienošana pie CentOS servera

Iekš CloudStack tika izveidota vēl viena virtuālā mašīna, kas būs serveris, kuram tiks pievienota bloku ierīce ar specifikācijām, kas redzamas 7.1. tabulā. Kad serveris ir izveidots, tad nepieciešams pārbaudīt, vai tas var sasniegt visas Ceph klastera mezglu ārējās IP adreses. Šim serverim nav nepieciešamības savienoties ar OSD iekšējām adresēm, jo tās ir paredzētas tikai OSD savstarpējai datu apmaiņai.

Kad jaunais serveris ir izveidots, tad nepieciešams no administrēšanas servera izveidot jaunu Ceph bloku sējumu, kuru varēs pievienot jaunizveidotajam serverim kā jaunu disku. Pirms tam ir jāpārliecinās, ka viss ir kārtībā ar klastera veselību, pēc kā var izveidot jaunu sējumu ar komandu `rbd create storage_volume –size 30720`, pēc kā tiks izveidots sējums ar 30720 MB (30 GB) izmēru un nosaukumu „storage_volume”.

Nosaukums	**Ārējā tīkla IP adrese, maska, vārteja**	**Publiskā IP adrese, maska, vārteja**	**Resursdatora nosaukums**
Klients	10.1.1.47, 255.255.255.0, 10.1.1.1	89.111.34.110, 255.255.255.0, 89.111.34.1	klients.hostnet.lv
Brīvpiekļuves atmiņa	**Procesors**	**SWAP**	**Operētājsistēmas disks**
1 GB	Intel Xeon E5645 @2.40GHz	1 GB	20 GB

7.1. tabula. **Klienta servera resursi, IP adreses un nosaukumi**

Iekš CentOS kodola nav iebūvēta `rbd` atbalsta, tāpēc nepieciešams uzstādīt pakotnes ar komandu `yum install ceph-common`. Tālāk nepieciešams pievienot ierīci uz klienta servera ar komandu:

```
echo "10.1.1.86,10.1.1.247,10.1.1.75
name=admin,secret=AQBJDF9V6L4nMBAAA1Gu7z0T9AaUcBSKx10uiw== rbd
storage_volume" > /sys/bus/rbd/add
```

, kur 3 IP adreses ir monitoru mezgliem, `secret` ir atslēga, ko var apskatīt uz kāda no monitoru mezgliem iekš faila `/etc/ceph/ceph.client.admin.keyring`. Un „storage_volume” ir iepriekš izveidotā sējuma nosaukums.

```
[root@klients ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1       20G  1,1G   18G   6% /
devtmpfs        490M     0  490M   0% /dev
tmpfs           495M     0  495M   0% /dev/shm
tmpfs           495M  6,5M  488M   2% /run
tmpfs           495M     0  495M   0% /sys/fs/cgroup
/dev/rbd0        30G   33M   30G   1% /home/ceph
```

7.2. attēls. **Piemontētie diski un to izmēri**

Tā rezultātā tiks iegūta jauna ierīce, kuru nepieciešams noformatēt par XFS failu sistēmu ar komandu `mkfs.xfs /dev/rbd0` , kur `rbd0` ir jaunizveidotā ierīce. Nepieciešams izveidot uz klienta servera jaunu direktoriju, kurai piemontēsim šo ierīci. Kad tas darīts, tad ar komandu `mount /dev/rbd0 /home/ceph/` , kur `/home/ceph` ir izveidotās direktorijas ceļš, piemontējam jauno disku [25]. Pēc tam pārliecināmies ar `df -h` komandu, ka darbība bijusi veiksmīga, un ir redzams piemontētais disks ar izmēru 30 GB, no kuriem pieejami ir 30 GB, kas ir redzams 7.2. attēlā.

Nepieciešams arī uzstādīt, lai ierīce tiktu piemontēta pie startēšanas, lai tas nebūtu jāveic katrreiz manuāli. Tā kā operētājsistēma ir CentOS 7, tad tiks izmantots systemd, lai pie servera startēšanas tiktu piemontēts disks automātiski. Lai to paveiktu, jāizveido fails `/etc/systemd/system/rbd-storage_volume.service`, kurš redzams 2. pielikumā. Kad fails ir izveidots, tad jāpalaiž šis serviss ar komandu `systemctl start rbd-storage_volume.service.` Ja viss ir kārtībā, tad to var pievienot pie servisiem, kas palaižas pie servera startēšanas ar komandu `systemctl enable rbd-storage_volume.service` [26].

8. TESTĒŠANA

Lielākais uzsvars šai nodaļā tiek likts uz uzticamības testēšanu un pārbaudi, kas notiek, kad serveri iziet no ierindas, atsakās darboties, kā arī, cik viegli vai sarežģīti ir paplašināt un samazināt diskus. Tiek skatīts, cik viegli var pievienot jaunu OSD serveri. Ar ātruma testēšanu neaizrāvos, jo testa vidē nav izveidota tāda konfigurācija, lai to veiktu, jo nav pieejami tādi resursi lai sasniegtu ievērības cienīgus rezultātus, tam būtu nepieciešami 10 gigabitu interneta pieslēgumi.

8.1. Bloku ierīces izmēru palielināšana

Ceph viena no galvenajām raksturīpašībām ir tā mērogojamība. Iepriekš izveidoto bloku ierīci, kas tika pievienota klienta serverim, var ļoti viegli palielināt, ja rodas tāda nepieciešamība un ir lieka diska vieta. To var izdarīt ar komandu `rbd --pool=rbd --size=61440 resize storage_volume`, kas palielinās šo sējumu no 30 GB uz 60 GB (komanda jāveic uz administrēšanas servera). Pēc tam jāpalaiž arī komanda `xfs_growfs /home/ceph` uz klienta servera, kas palielina failu sistēmu. Ar komandu `df -h` nepieciešams pārliecināties, ka diska izmēru maiņa ir bijusi veiksmīga un ka tagad apjoms ir 60 GB iepriekšējo 30 GB vietā.

```
[root@klients ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1       20G  1,1G   18G   6% /
devtmpfs        490M     0  490M   0% /dev
tmpfs           495M     0  495M   0% /dev/shm
tmpfs           495M  6,5M  488M   2% /run
tmpfs           495M     0  495M   0% /sys/fs/cgroup
/dev/rbd0        60G   34M   60G   1% /home/ceph
```

8.1. attēls. **Piemontētie diski un to izmēri pēc bloku ierīces izmēru palielināšanas**

Šīs komandas izvads ir redzams 8.1. attēlā. Ceph izmanto replikāciju, kura ir konfigurēta, lai katram blokam būtu 2 kopijas, tātad 60 GB šīs failu sistēmas reāli aizņems 120 GB uz klastera tad, kad būs pilna, tātad maksimālais izmērs failu sistēmai, kāda uz esošā klastera var atrasties, ir 75 GB.

8.2. Jauna OSD servera pievienošana

Dažādu iemeslu dēļ, var rasties vajadzība pēc papildus diska vietas, kuru var palielināt, izveidojot papildus OSD mezglus. Šeit tiks apskatīts, kā un cik vienkārši to ir izdarīt. Šim nolūkam tika izveidots vēl viens serveris, kura IP adreses un specifikācija ir redzama 8.1. tabulā. Viss ir analogi pirmajiem 3 OSD mezgliem, operētājsistēma arī ir CentOS 7. Arī uzstādīšana ir tieši tāda pati., uzstādot visus atjauninājumus un NTP servisu, pievienojot Ceph lietotāju un sagatavojot gan OSD, gan žurnāla diskus.

Nosaukums	**Ārējā tīkla IP adrese, maska, vārteja**	**Iekšējā tīkla IP adrese, maska, vārteja**	**Publiskā IP adrese, maska, vārteja**	**Resursdatora nosaukums**	
OSD 4	10.1.1.123, 255.255.255.0, 10.1.1.1	10.1.2.215, 255.255.255.0, 10.1.1.1	89.111.34.112, 255.255.255.0, 89.111.34.1	osd4.hostnet.lv	
Brīvpiekļuves atmiņa	**Procesors**	**SWAP**	**Operētājsistēmas disks**	**Žurnāla disks**	**OSD disks**
2 GB	2x Intel Xeon E5645 @2.40GHz	2 GB	20 GB	10 GB	50 GB

8.1. tabula. **OSD 4 servera resursi, IP adreses un nosaukumi**

Tālāk ir nepieciešams uzstādīt Ceph uz šī mezgla, ko veic palaižot komandu `ceph-deploy install osd4` no administrēšanas mezgla. Pēc Ceph uzstādīšanas, nepieciešams izveidot OSD un žurnāla diskus. Pirmkārt, ar `ceph-deploy disk zap osd4:xvdc` komandu izdzēšam visu disku saturu. Otrkārt, ar komandu `ceph-deploy osd create osd4:xvdc:/dev/xvdb1` pievienojam OSD disku, pēc kā tiks izvadīts, ka OSD 4 ir gatavs izmantošanai. Pēdējais, kas jāveic, ir konfigurācijas un administrēšanas atslēgu pārkopēšana uz šī servera ar komandu `ceph-deploy admin osd4`.

Ar komandu `ceph status` nepieciešams pārliecināties, ka jaunais OSD mezgls ir

pievienojies klasterim un gatavs darbam, komandas izvads ir redzams 8.2. attēlā, kur redzams, ka šobrīd pieejami ir 199 GB iepriekšējo 149 GB vietā, jo OSD 4 servera izmērs ir 50 GB, kā arī sadaļā `osdmap` ir redzams, ka nu jau ir 4 OSD.

```
[ceph@manage ceph-deploy]$ ceph status
    cluster 00d51eb2-c068-45d7-a098-c960ff7abd2e
     health HEALTH_OK
     monmap e1: 3 mons at {mon1=10.1.1.86:6789/0,mon2=10.1.1.247:6789/0,mon3=10.1.1.75:6789/0}
            election epoch 8, quorum 0,1,2 mon3,mon1,mon2
     osdmap e92: 4 osds: 4 up, 4 in
      pgmap v386: 64 pgs, 1 pools, 20128 kB data, 40 objects
            225 MB used, 199 GB / 199 GB avail
                  64 active+clean
```

8.2. attēls. **Komandas "ceph status" izvads pēc jauna OSD pievienošanas**

8.3. Jauna OSD diska pievienošana

Dažādu iemeslu dēļ, var rasties vajadzība pēc papildus diska vietas, kuru var palielināt, pievienojot OSD serveriem papildus OSD diskus, jo uz katra servera var būt vairāki diski. Pirmkārt, nepieciešams iekš CloudStack izveidot 2 diskus – 50 GB SATA disku un 10 GB SAS disku. Kad tas darīts, tad abus diskus pievieno OSD 4 serverim, lai tos varētu sagatavot darbam. Pirmkārt, ar parted rīku tiek izveidotas GPT partīciju tabulas abiem diskiem analogi iepriekš veidotajiem. Otrkārt, tiek noformatēts OSD disks, lai tam būtu XFS formāts. Žurnāla diskam tiks izmantots neformatēts disks. Tālāk no administrēšanas mezgla ar `ceph-deploy disk zap osd4:xvde` komandu tiek izdzēsts viss diska saturs, pēc tam tiek pievienots jauns OSD disks ar komandu `ceph-deploy osd create osd4:xvde:/dev/xvdf1`, kur `xvde` ir nesen pievienotais OSD disks un `/dev/xvdf1` ir jaunais žurnāla disks. Pēc komandas izpildes ir redzams izvads par veiksmīgu OSD

```
[ceph@manage ceph-deploy]$ ceph osd stat
     osdmap e4723: 5 osds: 5 up, 5 in
[ceph@manage ceph-deploy]$ ceph osd tree
ID WEIGHT  TYPE NAME          UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 0.24994 root default
-2 0.04999     host osd1
 3 0.04999         osd.3           up  1.00000          1.00000
-3 0.04999     host osd2
 1 0.04999         osd.1           up  1.00000          1.00000
-4 0.04999     host osd3
 2 0.04999         osd.2           up  1.00000          1.00000
-5       0     host localhost
-6 0.09998     host osd4
 0 0.04999         osd.0           up  1.00000          1.00000
 4 0.04999         osd.4           up  1.00000          1.00000
```

8.3. attēls. **5 OSD uz 4 serveriem**

pievienošanu, pēc kā ar komandu `ceph -w` var sekot līdzi datu pārbalansēšanai, jo ir pienācis klāt jauns OSD disks [30].

Šobrīd ir 4 OSD serveri un 5 OSD diski, kas redzams 8.3. attēlā. Ar `komandu ceph osd tree` ir redzams, kuram serverim pieder kuri OSD. Tikko pievienojām OSD 4 serverim papildus OSD disku, redzams, ka serverim osd4 šobrīd ir 2 OSD diski – osd.0 un osd.4.

8.4. OSD diska noņemšana

Nepieciešamības gadījumā arī iespējams samazināt klastera disku skaitu, noņemot no tā gan veselu OSD serveri, gan atsevišķi tikai vienu OSD disku. Pirmkārt, no administrēšanas mezgla nepieciešams norādīt, ka OSD tiek izņemts no klastera ar komandu `ceph osd out osd.4`, kas šai gadījumā izņem tikko pievienoto OSD disku. Tas jādara, lai Ceph var pārkopēt datus uz citiem OSD un pārbalansēt sevi. Pēc tam ir nepieciešams pieslēgties OSD serverim, uz kura ir osd.4 disks un tas ir jāapstādina ar komandu `service ceph stop osd.4`. Tālāk nepieciešams aizvākt OSD no CRUSH kartes, lai tam vairāk netiktu piegādāti dati ar komandu `ceph osd crush remove osd.4`, kā arī jādzēš OSD autentifikācijas atslēga ar komandu `ceph auth del osd.4`, visbeidzot var dzēst pašu OSD ar komandu `ceph osd rm osd.4` [30]. Kad OSD dzēsts, var atkal pārliecināties ar komandu `ceph status`, ka šobrīd ir aktīvi tikai 4 OSD. Ja kāds disks tiek neatgriezeniski bojāts, tad vispirms tas ir jānoņem, jāievieto tā vietā jaunais un jāpievieno klasterim analogi kā tas darīts 8.3. nodaļā.

8.5. Tīkla pārrāvumi

Katram OSD ir 2 mainīgi statusi – in vai out, kā arī up vai down. In vai out norāda vai OSD pieder klasterim vai arī nē, savukārt up vai down norāda vai OSD dēmons strādā vai arī nē. Testēšanas gaita ir sekojoša: uz kāda no serveriem palaiž skriptu, kas simulē 3 minūtes garu tīkla pārrāvumu ar skriptu, kurš izslēdz operētājsistēmas tīkla servisu:

```
#!/bin/sh
service network start
sleep 180
```

```
service network stop
```
Tikmēr no administrēšanas mezgla var vērot Ceph statusu ar komandu `ceph -w`, kura rāda klastera darbību un kas tiek izdarīts tīkla pārrāvuma laikā.

Ar šo komandu iespējams redzēt, ka pēc neilga brīža pārējie OSD ziņo, ka OSD, uz kura tika palaists skripts, ir atteicies (tā statuss mainās no up uz down). Kad internets tiek ieslēgts, tad redzams, ka klasteris atgriežas veselīgā stāvoklī (up). Tika mēģināts arī rakstīt diskā 1 GB datus un rakstīšanas brīdī izslēgt kāda mezgla tīklu, dati netika ietekmēti, tikai rakstīšanas ātrums, jo bija palikuši mazāk OSD, kur rakstīt. Ja tīkls tiek atslēgts uz ilgāku laiku, tad šī mezgla statuss tiek nomainīts no in uz out, kas nozīmē, ka tas ir ārpus klastera. Tai brīdī Ceph migrē visas izvietojuma grupas (PG) uz cietiem OSD un CRUSH algoritms tur datus vairs nerakstīs. Kad servera tīkls tika ieslēgts atpakaļ, tas automātiski pievienojās klasterim un atkal Ceph pārmigrēja visas izvietojuma grupas.

Tika mēģināts arī atslēgt gan vienu, gan divus monitora serverus reizē, tas neietekmēja klastera darbību, datus varēja turpināt lasīt un rakstīt. Kad mezgli atgriezās, tie veiksmīgi un automātiski pievienojās klasterim, nebija nepieciešama manuāla iejaukšanās.

8.6. Serveru pārstartēšana

Kad OSD serveris tiek pārstartēts, tas pats uzreiz atzīmē savu serveri kā neejošu (down). Savukārt, kad serveris ir startējies, tad visas izvietojuma grupas tiek atkal sinhronizētas un klastera veselība ir atgriezta. Šis process neradīja problēmas datu rakstīšanai, to varēja turpināt darīt ar mazāku ātrumu.

Monitora serveru pārstartēšana ir identiska tīkla pārrāvumiem. Pēc pārstartēšanas, visi servisi uz mezgla ieslēdzas un tas automātiski pievienojas klasterim. Jāpiebilst, ka sanāca vienreiz pārstartēt serveri, kurā nebija ieslēgts NTP serviss, kas izraisīja laika nobīdi par 0.06 sekundēm, kas ir vairāk kā Ceph atļautais 0.05 sekundes. Tad šis mezgls netika iekļauts klasterī līdz nebija veikta laika sinhronizācija uz tā. Kad to izdarīju, uz mezgla veicu Ceph dēmona restartu, tad tas atkal automātiski pievienojās klasterim.

8.7. Strāvas vada izraušana

Tā kā serveri, uz kuriem uzstādīts Ceph, nav fiziski serveri, tad simulēju šo darbību ar

„Force stop” opciju iekš CloudStack. Kad serveris tiek šādi izslēgts, tad tas pats atzīmē sevi kā neejošu (down). Kad tas tiek startēts, tad viss notiek analogi pārstartēšanas gadījumā. Ja serveris ir izslēgts ilgāku laiku, tad šī mezgla statuss tiek nomainīts no in uz out, kas nozīmē, ka tas ir ārpus klastera. Tai brīdī Ceph migrē visas izvietojuma grupas (PG) uz cietiem OSD un CRUSH algoritms tur datus vairs nerakstīs. Kad serveris tiek ieslēgts atpakaļ, tas automātiski pievienojās klasterim un atkal Ceph pārmigrē visas izvietojuma grupas. Ar monitora mezgliem viss notika analogi pārstartēšanas gadījumā, klastera darbība tika automātiski atgriezta.

8.8. Datu pieejamība no vairākiem serveriem

Šī brīža konfigurācijā pie klienta servera ir piemontēts viens disks. Šo disku nevar vienlaicīgi piemontēt arī pie cita servera un vienlaicīgi tajā rakstīt, tam ir nepieciešama failu sistēma, kas paredzēta klasteriem, kā piemēram ocFS2 vai GFS2. Šāda nepieciešamība varētu rasties, lai nodrošinātu augstu pieejamību piekļūšanai piemontētajai datu partīcijai. Piemēram, kāds slēdzas ar FTP klāt šim klients.hostnet.lv serverim, bet serveris nav pieejams, tad dati arī nebūs pieejami, lai gan pats glabātuves klasteris darbosies veiksmīgi. Šīm vajadzībām ir iespējams izveidot papildus serveri, piemēram, ar nosaukumu klients2.hostnet.lv un konfigurēt to kā pasīvo mezglu, kurš, 1.servera kļūmes gadījumā, kļūst aktīvs un pārņem vadību – tiek nomontēta partīcija no servera, kurā radās kļūda, piemontēta pie tā, kurš bija pasīvs. Cilvēkiem būtu jāslēdzas klāt virtuālajai IP adresei, kas tālāk pārvirzītu tos uz aktīvo klientu. Šādu konfigurāciju iespējams panākt ar keepalived rīku [31].

9. MONITORINGS

Uz klienta servera tika uzstādīts arī „Zabbix” monitoringa serveris, kuru pieslēdzu Ceph klasterim, lai varētu sekot līdzi klastera darbībai. „Zabbix” ir atklātā pirmkoda pieejamības un veiktspējas monitoringa risinājums. Tas piedāvā advancētu monitoringu, kļūdu ziņošanu, dažādus grafikus un citas lietas. Servera programmatūru atbalsta Linux, Solaris, HP-UX, AIX, FreeBSD, OS X. Savukārt, aģentus (serviss, kas iegūst informāciju par servera darbību un nosūta to uz „Zabbix” serveri) atbalsta gandrīz visas operētājsistēmas. „Zabbix” ir Latvijā balstīta kompānija [32].

Items	OSD 1
Ceph active MON	3
Ceph Operation	0 op/s
Ceph OSD in %	100 %
Ceph OSD up %	100 %
Ceph PG active	128
Ceph PG backfill	0
Ceph PG clean	128
Ceph PG creating	0
Ceph PG degraded	0
Ceph PG degraded %	0 %
Ceph PG down	0
Ceph PG incomplete	0
Ceph PG inconsistent	0
Ceph PG peering	0
Ceph PG recovering	0
Ceph PG remapped	0
Ceph PG repair	0
Ceph PG replay	0
Ceph PG scrubbing	0
Ceph PG splitting	0
Ceph PG stale	0
Ceph PG total	128
Ceph PG wait-backfill	0
Ceph rados free space	191.03 GB
Ceph rados total space	199.9 GB
Ceph rados used space	8.86 GB
Ceph Read Speed	0 B/s
Ceph Write Speed	0 B/s
ICMP loss	0 %
ICMP ping	Up (1)
ICMP response time	0.4ms
number of ceph-osd processes	1

9.1. attēls. **Zabbix monitorēšanas priekšmeti**

Zabbix serverim tika pielikts Ceph spraudnis [33], kas palīdz monitorēt Ceph klasterus. Priekšmeti, kas tiek vēroti klasterī, ir redzami 9.1. attēlā. Spraudņa uzstādīšanai bija nepieciešams pievienot šablonus, kas paredzēti tieši Ceph, kā arī uz Ceph mezgliem uzstādīt Zabbix aģentu, skriptu un konfigurācijas failu no iepriekš minētā spraudņa. 9.1. attēlā var redzēt, ka klasterī tiek monitorēts, cik ir aktīvie monitora mezgli, cik operācijas sekundē notiek, cik procentu no visiem OSD ir up un in režīmā, kā arī tiek sekots līdzi ceph PG stāvokļiem un RADOS diska vietai. Zemāk redzamie rādītāji, kas sākas ar ICMP ping, ir paredzēti katram mezglam atsevišķi, tie pārbauda to, vai mezgls atbild un pēdējais (*number of ceph-osd processes*) parāda, cik OSD procesu šobrīd ir uz OSD servera (ja uz servera ir vairāki OSD diski, tad tik arī būs procesu). Šos šablonus nepieciešams uzstādīt uz visiem klastera mezgliem, lai tie var ziņot par klastera stāvokli, kā arī individuāli paši par sevi.

10. REZULTĀTI

Darba gaitā ieguvu neaizvietojamu pieredzi un zināšanas, pētot un uzstādot dalītās paralēlās bojājumpiecietīgās failu sistēmas. Apguvu arī CloudStack lietotāja saskarni un izpratu, kā tas strādā, kā izveidot jaunus serverus, pievienot tiem tīklus un publiskās IP adreses.

Tika veiksmīgi izveidots Ceph klasteris, kas sastāv no 7 serveriem, serveris mezglu administrēšanai un klienta serveris, pie kā piemontēt failu sistēmu, kura datus glabā attālināti Ceph glabātuvē. Tika veikti arī dažādi testi, lai pārbaudītu klāstera darbību dažādu problēmu gadījumā, kā piemēram serveru pārstartēšana, tīkla problēmas, elektrības problēmas. Tika veikta arī papildus OSD pievienošana, un noņemšana, lai saprastu, cik šis process ir viegls vai sarežģīts, kā notiek mērogošana. Klasterim arī uzstādīta Zabbix monitoringa sistēma, lai varētu pārraudzīt tā stāvokli.

Rezultātā tika veiksmīgi izpildīti darbā aprakstītie veicamie uzdevumi un sasniegts darba mērķis - izveidots klasteris testam ar pētāmo failu sistēmu un izprasta tā darbība.

SECINĀJUMI

Pēc bakalaura darba izstrādes tika izdarīti šādi secinājumi:

- CloudStack ir visērtākais veids, ar kuru esmu strādājis jaunu serveru izveidošanai. To visu var izdarīt pāris sekunžu laikā, kas nav iespējams ar parastiem serveriem vai ar hipervīzoriem bez mākoņu platformām.
- Strādājot ar CentOS 7, ir jūtamas atšķirības, salīdzinot ar iepriekšējām CentOS versijām, ir jāapgūst dažas jaunas komandas un servisi.
- Pētītās failu sistēmas ir mūsu nākotne, jau daudz lielu uzņēmumu to lieto produkcijā, jo tas ir drošākais datu uzglabāšanas veids. Kā populārākos var minēt CERN un DreamHost.
- Ceph klasteris ir ļoti viegli paplašināms vai samazināms vajadzības gadījumā, kas ir ļoti noderīgi, jo beidzoties krātuvei, var rasties nepieciešamība pēc palīdzības.
- Ceph dokumentācija ir ļoti labi uzrakstīta un nav novecojusi, gandrīz visu nepieciešamo var atrast iekš tās, visas komandas ir ļoti labi izskaidrotas.
- Ceph ir arī ļoti drošs un automatizēts, jo, kāda mezgla kļūmes vai atteices gadījumā, Ceph visu izdara – izslēdz kļūdaino serveri no klāstera un turpina darboties, kāds arī ir viens no Ceph galvenajiem mērķiem.
- Ceph bloku ierīci var arī izmantot kā drošu failu glabātuvi, piemontējot to jebkuram serverim kā atsevišķu partīciju, kurā var rakstīt jebkādus datus.
- Ceph bloku ierīci var arī izmantot kā virtuālo mašīnu glabātuvi ar KVM vai QEMU hipervīzoriem. Drīzumā šī opcija būs pieejama arī XenServer hipervīzoriem, tādējādi Ceph ir labs kandidāts virtuālo mašīnu glabātuvei manā uzņēmumā (šobrīd tiek izmantoti NetApp produkti), jo uzņēmumā tiek izmantots XenServer.
- Ceph ir paredzēts gan lieliem, gan mazākiem uzņēmumiem, jo tas ir stipri mērogojams līdz pat vairākiem petabaitiem, kā arī tas ir atvērtā pirmkoda un nav nepieciešama speciāla aparatūra.
- Zabbix ir ļoti ērta monitoringa programmatūra, kuru var pielāgot visdažādākajām vajadzībām – pat Ceph klāstera uzraudzībai.

RESURSI

1. Wikipedia. List of file systems. [tiešsaiste] – [atsauce 03.03.2015]. Pieejams: http://en.wikipedia.org/wiki/List_of_file_systems#Distributed_parallel_fault-tolerant_file_systems

2. Hostnet. Cloud self service. [tiešsaiste] – [atsauce 03.03.2015]. Pieejams: http://www.hostnet.lv/lv/cloud-self-service/pakalpojums

3. Ceph documentation. Ceph block device. [tiešsaiste] – [atsauce 03.03.2015]. Pieejams: http://docs.ceph.com/docs/master/rbd/rbd/

4. Wikipedia. Distributed file systems. [tiešsaiste] – [atsauce 06.03.2015]. Pieejams: http://en.wikipedia.org/wiki/Clustered_file_system#Distributed_file_systems

5. Samuel Lang. Parallel File Systems. [tiešsaiste] – [atsauce 06.03.2015]. Pieejams: http://www.cs.iit.edu/~iraicu/teaching/CS554-F13/lecture17-pfs-sam-lang.pdf

6. Techopedia. Fault tolerance. [tiešsaiste] – [atsauce 06.03.2015]. Pieejams: http://www.techopedia.com/definition/3362/fault-tolerance

7. Benjamin Depardon, Gael Le Mahec, Cyril S´eguin. Analysis of Six Distributed File Systems. [tiešsaiste] – [atsauce 06.03.2015]. Pieejams: https://hal.inria.fr/hal-00789086/file/a_survey_of_dfs.pdf

8. Ceph. Ceph architecture. [tiešsaiste] – [atsauce 12.03.2015]. Pieejams: http://ceph.com/docs/master/architecture/

9. Sage Weil, Scott Brandt. CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data. [tiešsaiste] – [atsauce 15.03.2015]. Pieejams: http://ceph.com/papers/weil-crush-sc06.pdf

10. Ceph. Monitor config reference. [tiešsaiste] – [atsauce 15.03.2015]. Pieejams: http://docs.ceph.com/docs/master/rados/configuration/mon-config-ref/

11. Slideshare. Ceph intro and architectural overview by Ross Turk. [tiešsaiste] – [atsauce 18.03.2015]. Pieejams: http://www.slideshare.net/buildacloud/ceph-intro-and-architectural-overview-by-ross-turk

12. Ceph. OSD Config refference – Scrubbing. [tiešsaiste] – [atsauce 20.03.2015]. Pieejams: http://docs.ceph.com/docs/master/rados/configuration/osd-config-ref/#scrubbing

13. Redhat. Ceph client architecture. [tiešsaiste] – [atsauce 20.03.2015]. Pieejams: https://access.redhat.com/beta/documentation/en/red-hat-ceph-storage-123-red-hat-ceph-architecture/chapter-2-ceph-client-architecture

14. Stack Overflow. What is copy-on-write? [tiešsaiste] – [atsauce 20.03.2015]. Pieejams: http://stackoverflow.com/questions/628938/what-is-copy-on-write

15. YouTube. Ceph Architectural overview by Ross Turk.[tiešsaiste] – [atsauce 25.03.2015]. Pieejams: https://www.youtube.com/watch?t=429&v=OyH1C0C4HzM

16. Pineight. Block device. [tiešsaiste] – [atsauce 30.03.2015]. Pieejams: http://pineight.com/ds/block/

17. Ceph. Ceph block device. [tiešsaiste] – [atsauce 30.03.2015]. Pieejams: http://ceph.com/docs/master/rbd/rbd/

18. Ceph. Ceph object gateway. [tiešsaiste] – [atsauce 30.03.2015]. Pieejams: http://ceph.com/docs/master/radosgw/

19. Ceph. Ceph filesystem. [tiešsaiste] – [atsauce 30.03.2015]. Pieejams: http://ceph.com/docs/master/cephfs/

20. Ceph. Hardware recommendations. [tiešsaiste] – [atsauce 05.04.2015]. Pieejams: http://docs.ceph.com/docs/master/start/hardware-recommendations/

21. Ceph. Journal config reference. [tiešsaiste] – [atsauce 05.04.2015]. Pieejams: http://ceph.com/docs/master/rados/configuration/journal-ref/

22. Virtualization blog. My adventures with Ceph Storage Part 3. [tiešsaiste] – [atsauce 05.04.2015]. Pieejams: http://www.virtualtothecore.com/en/adventures-ceph-storage-part-3-design-nodes/

23. Ceph. Quick start preflight. [tiešsaiste] – [atsauce 08.04.2015]. Pieejams: http://ceph.com/docs/master/start/quick-start-preflight/

24. Ceph. Network configuration reference. [tiešsaiste] – [atsauce 10.04.2015]. Pieejams: http://docs.ceph.com/docs/master/rados/configuration/network-config-ref/

25. Ceph Block device quick start. [tiešsaiste] – [atsauce 25.04.2015]. Pieejams: http://ceph.com/docs/master/start/quick-rbd/

26. Virtualization blog. My adventures with Ceph Storage Part 6. [tiešsaiste] – [atsauce 27.04.2015]. Pieejams: http://www.virtualtothecore.com/en/adventures-with-ceph-storage-part-6-mount-ceph-as-a-block-device-on-linux-machines/

27. Ceph. Block device commands. [tiešsaiste] – [atsauce 01.05.2015]. Pieejams: http://ceph.com/docs/master/rbd/rados-rbd-cmds/

28. InterWorx. CentOS7 released: Here's what's new. [tiešsaiste] – [atsauce 03.05.2015]. Pieejams: http://www.interworx.com/community/centos-7-released-heres-whats-new/

29. Linode. Network iptables. [tiešsaiste] – [atsauce 05.05.2015]. Pieejams: https://www.linode.com/docs/security/firewalls/control-network-traffic-with-iptables

30. Ceph Adding/removing OSDs. [tiešsaiste] – [atsauce 08.05.2015]. Pieejams: http://ceph.com/docs/master/rados/operations/add-or-rm-osds/

31. Virtualization blog. My adventures with Ceph Storage Part 8. [tiešsaiste] – [atsauce 12.05.2015]. Pieejams: http://www.virtualtothecore.com/en/adventures-with-ceph-storage-part-8-veeam-clustered-repository/

32. Zabbix. What is Zabbix. [tiešsaiste] – [atsauce 12.05.2015]. Pieejams: http://www.zabbix.com/product.php

33. GitHub. Zabbix plugin for Ceph monitoring. [tiešsaiste] – [atsauce 12.05.2015]. Pieejams: https://github.com/thelan/ceph-zabbix

PIELIKUMI

1.pielikums. **Ceph konfigurācijas fails „ceph.conf”**

```
[global]
fsid = 00d51eb2-c068-45d7-a098-c960ff7abd2e
mon_initial_members = mon1, mon2, mon3 # norāda, kuri ir monitora mezgli
mon_host = 10.1.1.86,10.1.1.247,10.1.1.75 # norāda to IP adreses
auth_cluster_required = cephx # aktivizē autentifikāciju
auth_service_required = cephx # aktivizē autentifikāciju
auth_client_required = cephx # aktivizē autentifikāciju
filestore_xattr_use_omap = true

public network = 10.1.1.0/24 # norāda ārējā tīkla CIDR
cluster network = 10.1.2.0/24 # norāda klastera (iekšējā) tīkla CIDR

#Choose reasonable numbers for number of replicas and placement groups.
osd pool default size = 2 # Write an object 2 times
osd pool default min size = 1 # Allow writing 1 copy in a degraded state
osd pool default pg num = 128
osd pool default pgp num = 128
#Choose a reasonable crush leaf type - 0 for a 1-node cluster.
#1 for a multi node cluster in a single rack and so on
osd crush chooseleaf type = 1
```

2. pielikums. **rbd-storage_volume.service systemd fails**

```
[Unit]
Description=”Bakalauram storage_device"
Conflicts=shutdown.target
Wants=network-online.target
After=NetworkManager-wait-online.service
[Service]
Type=oneshot
ExecStart=/sbin/modprobe rbd
ExecStart=/bin/sh -c "/bin/echo 10.1.1.86,10.1.1.247,10.1.1.75
name=admin,secret=AQBJDF9V6L4nMBAAA1Gu7z0T9AaUcBSKx10uiw== rbd
storage_volume" > /sys/bus/rbd/add"
ExecStart=/bin/mount /dev/rbd0 /home/ceph
TimeoutSec=0
RemainAfterExit=yes
[Install]
WantedBy=multi-user.target
```

Bakalaura darbs “Dalītās paralēlās bojājumpiecietīgās failu sistēmas” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā versija atbilst izdrukai.

Autors:_____ Eduards Kļaviņš

Rekomendēju/nerekomendēju darbu aizstāvēšanai (nevajadzīgo izsvītrot)
Vadītājs: Dr.dat. Leo Trukšāns _____

Darbs iesniegts Datorikas fakultātē

Dekāna pilnvarotā persona:

Darbs aizstāvēts bakalaura darbu komisijas sēdē

Komisija: