Tartu Ülikool
Arvutiteaduse instituut
Informaatika õppekava

Ott Eric Oopkaup

Tartu Ülikooli Teadusarvutuste keskuse andmemassiivi ülesseadmine ja jõudluse testimine

Bakalaureusetöö (9 EAP)

Juhendaja: Alo Peets

Tartu 2024

Tartu Ülikooli Teadusarvutuste keskuse andmemassiivi ülesseadmine ja jõudluse testimine

Lühikokkuvõte:

Tartu Ülikooli Teadusarvutuste keskus pakub arvutusvõimsust ja andmehoidlat sadadele teadlastele. Suurenevate andmehulkade mahutamiseks on olulisemaks muutunud failisüsteemide peenhäälestus ja profileerimine, et garanteerida failisüsteemi jõudlus arvutuskeskude töövoogude jooksutamisel. Käesoleva lõputöö teoreetiline eesmärk on kirjeldada suure jõudlusega salvestusmassiivi seadistamist, profileerimist ja jõudluse testimist.

Lõputöö selgitab erinevaid mõõdikuid ja parameetreid failisüsteem seadistamisel ning sisaldab andmeid Lenovo DSS-G põhise failisüsteemi jõudluse testimise tulemustest. Lõputöö praktiliste tegevuste käigus oli eesmärgiks Lenovo DSS-G251 andmehoidla dokumentatsiooniga tutvumine, juhtserverite ülesseadistamine, määrata failisüsteemi metaandmete- ja andmeplokkide suurused garanteerimaks kohane jõudlust andmemassiivile ning testida lõppseadistuses töötava süsteemi jõudlust.

Võtmesõnad: Linux, failisüsteem, andmete hoiustamine, kõvaketas, jõudlustestimine, häälestamine

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Configuring and performance benchmarking a storage array for the University of Tartu HPC Center

Abstract:

The University of Tartu HPC Center offers computational power and data storage to hundreds of researchers. In order to accommodate growing amounts of data, fine-tuning and profiling file systems meant for HPC use cases has become essential. The theoretical aim of this thesis is to describe the setup, profiling and benchmarking of a high performance file system.

The thesis explains the various metrics and characteristics of setting up a filesystem and also includes performance results on multiple different configurations of a Lenovo DSS-G based filesystem. The practical aim of the thesis was to familiarize with the documentation of the Lenovo DSS-G251 storage array,

set up the servers controlling the filesystem, determine the file system metadata and data block sizes in order to guarantee the necessary reliability of the file system and to test the final performance of the storage array.

Keywords: Linux, file system, data storage, hard disk, benchmarking, tuning

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

Sissejuhatus

Tartu Ülikool on Eesti vanim ja kõrgeimalt hinnatud ülikool, kus õpib umbes 15 000 tudengit ja neid õpetab 2000 akadeemilist töötajat. Lisaks õppetööle tehakse Tartu Ülikoolis ka palju teadustööd ning digitaliseerimise tulemusena on enamik teadustööd andmemahukas. Kasvavate andmemahtude juures on teadlastel üha enam tekkinud vajadus suuremate arvutisüsteemide vastu kus andmeid töödelda. Probleemi lahendamiseks asutati 2009. aastal Tartu Ülikooli Teadusarvutuste keskus.

Tänaseks on Teadusarvutuste keskuse andmehoidlatest hoiustatavate andmete hulk kasvanud 9000 terabaidini ning kasvavate andmemahtude juures on pidev vajadus uute kettamassiivide järgi. Sealhulgas on tingitud ka vajadus testida ning profileerida andmemassiivide töö eesmärgiga garanteerida ka piisav jõudlus HPC töövoogude jooksutamiseks.

Käesoleva bakalaureusetöö eesmärgiks on Teadusarvutuste keskusele soetatud andmemassiivi juhtserveritele tarkvara paigaldamine, seadistamine, olemasoleva infrastruktuuriga sidumine ning ühildamine kui ka andmemassiivi häälestamine. Lisaks jooksutatakse komplekt jõudlusteste eri parameetritega mille ülesandeks on failisüsteemi töö profileerimine, et langetada teadlik otsus failisüsteemi lõpp-parameetrite kohta.

Käesolevas lõputöös on kolm peatükki, millest esimeses kirjeldatakse failisüsteemide testimise ja profileerimise metoodikat ning parameetreid, mida saab muuta koos lühiülevaatega testitavast andmemassiivist. Teises peatükis vaadeldakse Lenovo DSS-G tüüpi andmemassiivi ülesseadmist ning samme, mis on vajalikud ühe failisüsteemi integreerimiseks Teadusarvutuste keskuse olemasoleva taristuga. Lõputöö viimases peatükis uuritakse lähemalt failisüsteemi jõudluse testimise metoodikat ja tulemusi, ning tutvustatakse võimalusi edasiste andmete kogumiseks, mis aitavad failisüsteemi jõudlust paremini määrata. Lisadena saab tutvuda väljatöötatud skriptidega millega jõudlust kontrolli ning näidetega failisüsteemi seadistamisest.

1. Mõisted ja terminid

DSS-G - *Lenovo Distributed Storage Solution*, Lenovo toodetud terviklahendus kettamassividest mis tarkvaraliselt kasutab GPFS-i

GPFS - *General Parallel File System,* IBM-i väljatöötatud failisüsteem ja kaasnev tarkvara

GNR - *GPFS Native Raid*, GPFS-i tarkvaraline RAID

Hajus failisüsteem - failisüsteem, mille salvestusressursid ja kliendid on arvutivõrgus hajutatud

HPC - High Performance Computing, kõrgjõudlusega andmetöötlus (keskus)

IOPS - Sisend-väljundoperatsiooni sekundis, maksimaalne lugemiste ja kirjutuste arv sekundis

Kettamassiiv - mitmetest kõvaketastest koosnev salvestussüsteem

LACP - Link Aggregation Control Protocol, IEEE 802.3ad võrgustandardi kanalite agregeerimise ohje protokoll

Plokk - Andmete ketta- ja lintseadmetele salvestuse ja sealt võtu üksus

RAID - Salvestustehnoloogia, mis ühendab mitu ketast või kettasektsiooni üheks loogiliseks üksuseks

2. Kettamassiivi lühitutvustus

Käesolevas peatükis antakse lühiülevaade kettamasiivist ning IBM-i väljatöötatud failisüsteemist nimega GPFS (General Parallel File System). Lisaks kirjeldatakse kettamassivi testimise parameetreid ning põhjuseid.

2.1 General Parallel File System

GPFS[1] on IBM-i väljatöötatud failisüsteem [1], mis toetab nii mitmeid eri tüüpi andmekandjaid, ühtlustab andmetele ligipääsu oma tarkvaraliste lahendustega. GPFS on hajus failisüsteem, ehk tarkvara pakub samaaegset ligipääsu failisüsteemile üle eri võrguprotokollide. Hajusa failisüsteemina võivad asuda riistvaralised komponendid eri asukohtades, kuid kasutajale näib failisüsteem ühe tervikuna.

GPFS-i alged jäävad 90. aastatesse, kui algselt oli tegu IBMi siseprojektiga nimega Tiger Shark, mille eesmärgiks oli skaleeritava failisüsteemi loomine [2]. Üle kahekümne aasta tarkvaraarendust ning paralleliseeruv disain on soosinud GPFS-i kasutust just kõrge intensiivsusega töövoogudes kus andmemassiivil asuvaid andmeid loetakse ning kirjutatakse paralleelselt mitmete sadade või tuhandete klientserverite poolt korraga. Teadusarvutuste keskuse juhataja Ivar Koppeli sõnul, on keskuses GPFS kasutuses olnud alates aastast 2012**.**

2.2 GPFS Native RAID

GNR (GPFS Native Raid) GPFS-i tarkvaraline RAID kiht, mis seob failisüsteemi moodustavad andmekandjad tarkvaraliselt kokku [3]. GNR toetab nii eri veaparanduskoode, läbivaid kontrollsummasid, kui ka täiustatud failipaigustusalgoritme, mille eesmärk on garanteerida failisüsteemi veataluvus ning jõudlus. Tarkvaralise RAID-i peamine tugevus on paindlikus veakontrolli taseme määramisel ning tarkvaraline andmevoo kontrollimine lihtsustab ka riistvaralist konfiguratsiooni. Lisaks seob GNR kogu kettamassiivi andmekandjad üheks *declustered array* tüüpi massiiviks, kus üksiku andmekandja vea puhul hajutatakse veakoodi taastamine kõikide andmekandjate vahel massiivis. [4]

[1] Ametlik tootenimi on alates 2023 IBM Storage Scale, nime GPFS kasutatakse failisüsteemi, kui tarkvara nimena

2.3 DSS-G salvestuslahendus

Lenovo Distributed Storage Solution for IBM® Storage Scale (DSS-G) on Lenovo ja IBM-i koostöös välja töötatud terviklik salvestuslahendus, mis toetub Lenovo riistvarale ning kasutab tarkvaralise lahendusena GPFS-i [5]. Lenovo riistvaralistest konfiguratsioonidest käsitleb lõputöö DSS-G2xy varianti, kus "x" määrab ära D3284 tüüpi HDD kettašasside arvu ning "y" D1224 SSD kettašasside arvu.

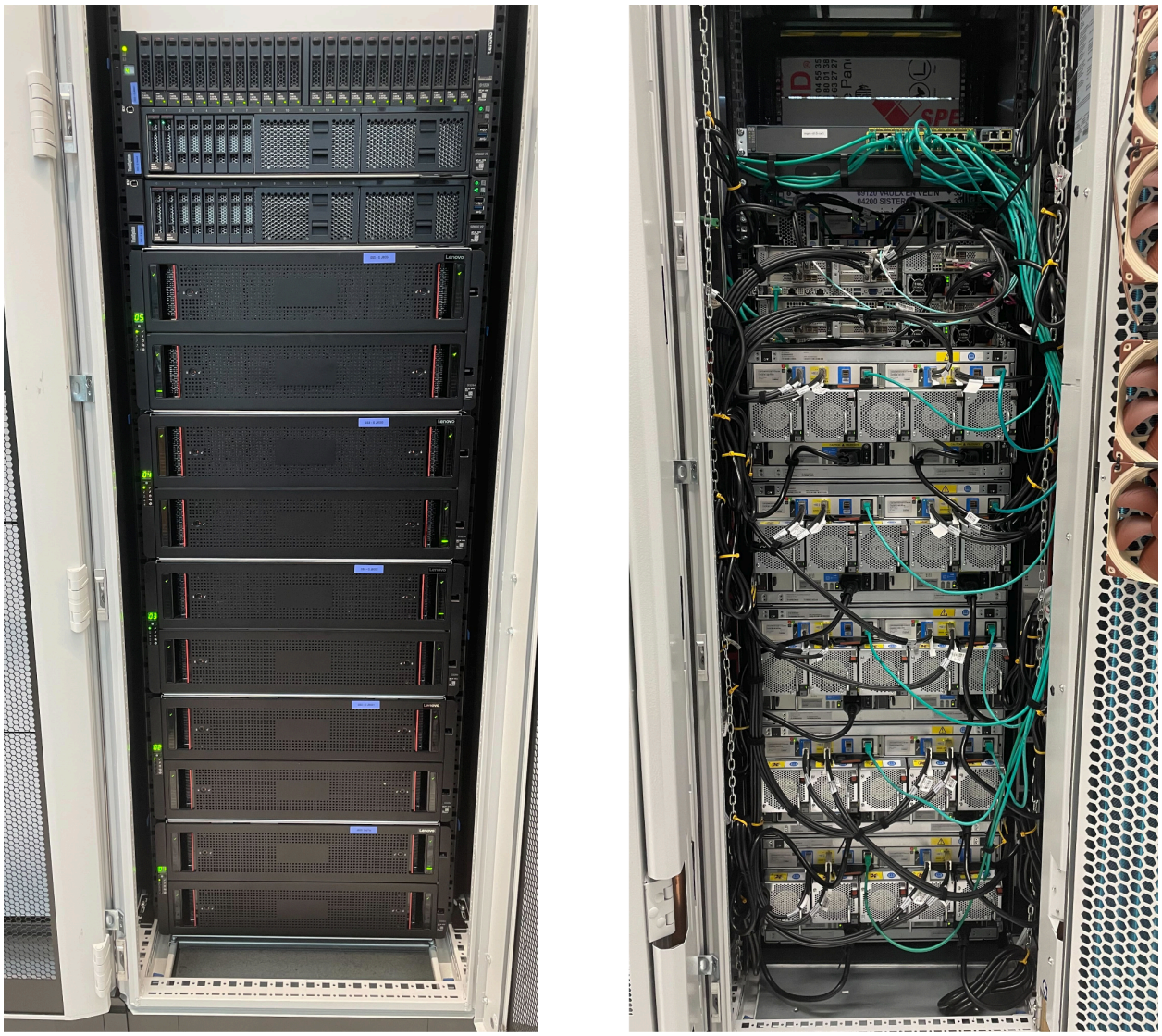
Tarkvaraliselt pakitakse salvestuslahendusega kaasa *IBM Storage scale,* mis sisaldab endaga kõike vajalikku DSS-G lahenduse ülesseadmiseks. Vastavas DSS-G tarkvaralises komplektis sisaldub nii juhtserverite püsivara (*firmware*), võrguadapterite püsivara ning GPFS.

2.4 HPC failisüsteemi tehnilised parameetrid

Käesolevas bakalaureusetöös käsitletakse Teadusarvutuste keskusele 2023. aastal välja kuulutatud riigihanke[2] korras soetatud salvestuslahendust, mudelinumbriga DSS-G251 ning mis koosneb viiest HDD kettakastist, ühest SSD tüüpi kettakastist ning kahest Lenovo SR650v2 tüüpi juhtserverist [6]. Joonisel 1 vasakpoolsest andmemassiivi eesvaatest on näha kõige üleval D1224 tüüpi SSD kettakandurit, mis hoiustab 24 tükki 7.68TB SSD kõvakettaid. Kettakandurist allpool asuvad kaks juhtserverit *sol1* ja *sol2* ning seadmekapi alumises osas asub viis D3284 tüüpi kettakandurit, millest igaüks hoiustab 84 tükki 20TB mahutavusega kõvaketast, kogumahutavusega 8400 terabaiti

Varasemalt on HPC kasutuses DSS-G260 andmemassiiv, kuid käesolevas töös kirjeldatud massiiv on esimene, mis sisaldab lahenduses ka SSD tüüpi kettakandurit mille eesmärgiks on nii failisüsteemi metaandmete kui ka SSD põhise saalimisala hoiustamine.

[2] Riigihanke viitenumber: 267390



Joonis 1. Foto töös käsitletavast andmemassiivist

2.5 Sisend-väljund tehinguid sekundis

IOPS (*Input-output operations per second*) on ühe andmekandja või failisüsteemi esimene omane parameeter. Oma olemuselt näitab IOPS ära, mitu tehingut sekundis saab andmekandjal sooritada ning läbi selle ka keskmise pöördusaja. IOPS on tugevalt seotud andmekandja tüübiga - kui pooljuhtketastel on pöördusaeg vähe varieeruv tänu elektroonilisele ülesehitusele , on pöörlevatel kõvaketastel latents seotud ka loendurnõela asukohaga vastavalt ketta plattrile. Tihtipeale eelistavad eri tüüpi failisüsteemid alustada kõvakettale andmete

kirjutamist just välisest servast, kuna võrdeliselt liigub pöörleval kettal välimises servas nõela alt läbi rohkem andmesektoreid. GPFS-i konfiguratsioon [7] lubab määrata failisüsteemi loomise ajal andmeplokkide paigutamise algoritmi, mis DSS-G lahenduse puhul jaotab andmeplokid suvaliselt eesmärgiga ühtlustada failisüsteemi jõudlus.

Peale tehingute arvu sekundis on oluline kaaluda ka ühe andmekandja reaktsiooniaega, mida tihtipeale mõõdetakse keskmise väärtusena. Lisaks on raske hinnata ühte andmekandjat ainult nende kahe väärtuse põhjal, kuna tavapärases IO-operatsiooni ahelas on nii puhvreid kui ka mitmeid lõimesi. Veel oleneb andmekandja jõudlus operatsioonisüsteemi sätestest ja eri töökoormuste karakteristikutest.

2.6 Kirjutamise ja lugemise andmeedastuskiirus

Sidudes operatsioonide arvule sekundis külge ka sooritatud tehingu suuruse, saame määrata ühe andmekandja andmeedastuskiiruse. Tavaliselt määratakse tehingu suurus failisüsteemi ploki suurusega, mis on operatsioonisüsteemi tasemel kõige väiksem tehing mida andmekandjale saab kirjutada või sealt lugeda. Suuremaid tehinguplokke saab ka puhverdada näiteks juhtudel, kus tehinguploki suurus on väiksem kui andmekanduril.

Andmeedastuskiiruse hindamiseks eristatakse ka andmekandjalt lugemist ja kirjutamist. Kuigi kõvaketastel on tihtipeale lugemine ja kirjutamine sama jõudlusega, ei pruugi SSD tüüpi ketastel see alati nii olla. Näiteks, SSD ploki lugemine on tavapäraselt üks tehing, aga kirjutamiseks peab SSD kontroller kontrollima enne välkmälu sisu ning vajadusel nullima ta enne andmete kirjutamist [8]. See võib tähendada mõnda aega kasutatud SSD ketta kirjutamisjõudluse langust ajas. Probleemi lahendamiseks saab tänapäevastel andmekandjatel lülitada sisse *TRIM*-i, mis nullib mäluraku sisu pärast andmete kustutamist.

2.7 Ligipääsumustrid andmekandjatel

Andmekandja reaktsioonikiirusel on määravav omadus ka reaktsiooniaja ühetaolisus. Kui pooljuhtkettad on tööpõhimõtetelt täiesti elektroonilised ning ühe mäluraku lugemise aeg on ühtlane, siis pöörleval kõvakettal väljendub reaktsiooniajas ka füüsilse liikuva komponendi (loendurnõela) rändeaeg. Kui kõvakettal sooritatavad andmetehingud asuvad järjestikustes sektorites ketta platteri peal, siis on tehingu aeg konstantne. Küll aga kui tehinguid ei tehta järjestikustes kettasektorites, peab iga tehingu jaoks magnetnõel liikuma.

Siinkohal saame eristada kahte peamist tüüpi ligipääsumustrit või töökoormust, milleks on siis järjestatud või juhuslik töökoormus. SSD tüüpi andmekandjatel väljenduvad üldiselt mõlemad töökoormused sarnase jõudlusega, kuid HDD tüüpi ketastel on juhusliku iseloomuga koormused palju väiksema jõudlusega.

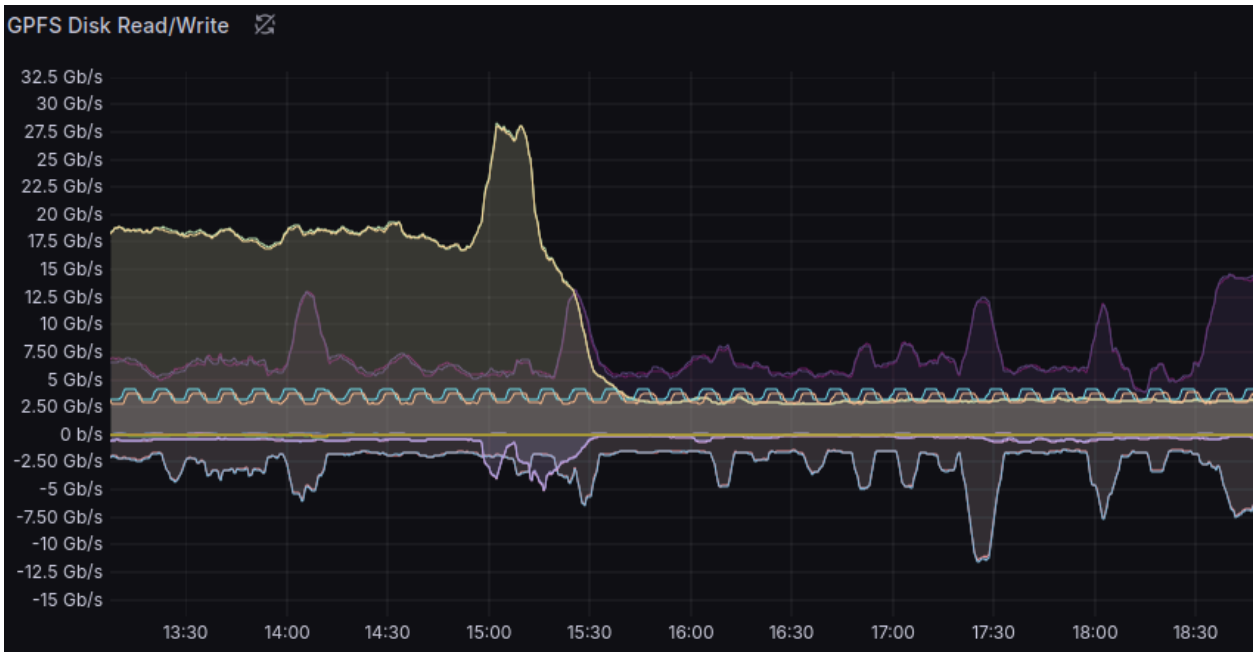
2.8 Failisüsteemi ploki suurus

Kuigi failisüsteemide testimise ja profileerimisel on kümneid peenhäälestatavaid parameetreid, siis DSS-G lahenduse juures on IBM ja Lenovo suurema osa juba ära häälestanud, aga lõppkasutajale on kaks suuremat muudetavat parameetrit- failisüsteemi andmeploki suurus ning failisüsteemi metaandmete ploki suurus. Lisaks, kuna seadistatavasse failisüsteemi kuulub 418 HDD tüüpi ning 24 SSD tüüpi andmekandjat, siis ei häälestata kettaid individuaalselt vaid tervikuna.

GPFS-i kontekstis on erinevus andme ja metaandmete plokis küllaltki ilmne. Nimelt, hoiustatakse ühe failiga seotud metaandmed nagu nimi, indekssõlm (*inode*), failiõigused ning failiga seotud andmeplokkide asukohad metaandmete hulgas ja andmeplokid hoiustavad faili sisu. See tähendab, et saame profileerida vähemalt kahe erineva komponendi jõudlust, mis on siis vastavalt seotud kas andmete lugemise-kirjutamisega või metaandmete muudatustega.

Kuna failisüsteemi andmeploki suurus määrab ka ära operatsioonisüsteemi tasemel kõike väiksema tehingu mida saab failisüsteemil teha, on tegu peamise parameetriga, mida kohendame käesoleva failisüsteemi seadistamisel.

Vaadeldes HPC sisemise monitooringu kogutud andmeid (Joonis 2), saame ka teha mõned järeldused igapäevase töökoormuse kohta ning kuidas see väljendub failisüsteemi ligipääsumustrites. Andmekiiruse graafikult saame lugeda, et peamisel HPC klastri failisüsteemil /gpfs/space on pidev kirjutamine ja lugemine suurusjärgus 10gbit/s (kombineeritult 20gbit/s), mis intensiivse kasutuse ajal võib ulatuda kuni 40 gbit/s (kombineeritult 80gbit/s).



Joonis 2. Väljavõte tavapärasest koormusest HPC failisüsteemidel

Päringute järgsuse uurimiseks tuleb kõigepealt analüüsida HPC arvutusklastri arhitektuuri. Nimelt kuulub klastrisse suurusjärgus 60 eri serverit, mis siis saalivad oma andmeid klastri failisüsteemi peal. Eeldades, et kõik 60 serverit teevad järjestikkuseid päringuid, on ilmne, et failisüsteem ise peab teenindama suvalise sisendjärjestusega erinevate klientserverite päringuid selles järjekorras, kuidas nad kontrollserveriteni jõuavad. Lisaks, kuna üks server ei pruugi piirduda 1 tuumalise arvutusega, kasvab keskmine paralleelsete päringute arv. Arvestades, et HPC ise ei reguleeri arvutusklastri peal tehtavaid töid, võib eeldada, et faile töötlevad programmid tulevad segamini nii järjestatud kui ka juhuslike andmepärimismustriga ning ei saa kindel olla ka programmi poolt operatsioonisüsteemilt küsitava päringu suuruses.

2.9 Töö eesmärgid

Käesoleva bakalaureusetöö eesmärgiks on salvestuslahenduse ülesseadmine, testimine ning profileerimine. Kuigi Teadusarvutuste keskusel on kasutuses kaks sarnast lahendust, muutub aja jooksul failisüsteemil käitavate tööde eesmärk ning iga salvestuslahenduse ülesseadmist koheldakse individuaalselt.

Testimise ning profileerimise peamine eesmärk on leida kõige paremini sobivad failisüsteemi ploki suuruse parameetrid, mis tagavad piisava jõudluse salvestuslahenduse planeeritud töö jaoks. Lisaks, kuigi sünteetilised testid ei ole üks ühele vastavad reaalelulisele jõudlusele, aitavad jõudlustestid süsteemi administraatoril määrata riistvara võimekuse.

Käesoleva bakalaureusetöö väljundiks on määrata failisüsteemi metaandmete- ning andmeploki suurused nii, et leida “kuldne kesktee” IOPS-i ning andmeedastuskiiruse vahel. Oma olemuselt on failisüsteemi ploki suurus oluline just seepärast, et parameeter määrab ära kõige väiksema päringu, mida operatsioonisüsteem andmekandja vastu saab sooritda. See tähendab, et liiga suure ploki ja väikese päringu suurusega loetakse failisüsteemilt liiga palju andmeid, mis tehislikult piirab andmemassiivi läbilaskevõimet. Küll aga on ploki suuruse sättimine väikeseks pärssiv, kuna liiga väikese andmeploki suuruse puhul peab failisüsteem tegema mitu päringut ühe suurema tehingu jaoks.

3. Failisüsteemi esialgne seadistamine

Järgmistes peatükkides käsitletakse protseduuri failisüsteemi esialgsest installeerimisest ning konfiguratsioonist. Selle osa eesmärgiks oli testida terviklahendust ning selle korrektsust peamiselt et rahuldada riigihanke üleandmise tingimus (süsteem toimib, allkirja saab anda).

3.1 Tarkvara Confluent paigaldus

Tarkvara paigaldus kahele juhtserverile toimub läbi Lenovo arendatud juhttarkvara Confluent. Confluent on mõeldud suuremahuliste Lenovo süsteemide haldamiseks nii tarkvara, seadmete draiverite kui ka operatsioonisüsteemi konfiguratsiooni jaoks. HPCs igapäeva töös Confluenti ei kasutata, kuid tarkvara on vajalik esialgseks operatsioonisüsteemi seadistamiseks ning seetõttu sai loodud virtuaalmasin kuhu Confluent paigaldati. Virtuaalmasinale seadistati ka vastavad võrguadapterid ühendusid olemasolevate HPC võrkudega ning läbi nende juhtida DSS-G tarkvara paigaldust.

Confluendi ülesseadmine vastavalt Lenovo juhendile [9] on üks võimalik viis, kuid DSS-G tarkvaralise korrektsuse mõttes paigaldati Confluent vastavalt DSS-G paigaldusjuhendile [10].

Esimeste sammudena sai Confluent Lenovo veebisaidilt alla laetud, .tar laiendiga arhiiv lahti pakitud ning kohalikust repositooriumist ka virtuaalmasinale paigaldatud.

```
tar xvJf confluent*.tar.xz
cd lenovo-hpc-el7
./mklocalrepo.sh
yum install -y lenovo-confluent confluent_osdeploy-x86_64 tftp-server

systemctl enable confluent --now
systemctl enable tftp.socket --now
systemctl enable httpd --now

#Laeme käsud edasi töötamiseks keskkonda sisse
source /etc/profile.d/confluent_env.sh
```

Järgmisena paigaldatakse Confluenti jaoks vajalikud lisapaketid ja teenused. Lisaks käivitatakse ka nimeserveri teenus dnsmasq ja kellaserveri teenus chronyd.

```
yum install -y rsync bc dnsmasq chronyd
systemctl enable dnsmasq chronyd --now
```

Pärast Confluenti teenuse kontrollimist, genereeritakse SSH teenuse jaoks vajalikud võtmed ning alustatatakse juhtserveritele paigaldatava tarkvara sisselaadimist.

```
ssh-keygen -t ed25519
osdeploy initialize -i

#Laeme sisse RedHat Enterprise Linux operatsioonisüsteemi tarkvara
osdeploy import /shared/iso/rhel-8.6-x86_64-dvd.iso

#Laeme sisse DSS-G tarkvara
mkdir -p /opt/lenovo/dss
cp dss-g-4.5a-advanced-5.1.tgz /opt/lenovo/dss/
cd /opt/lenovo/dss
tar xvzf dss-g-4.5a-advanced-5.1.tgz

#Pakime lahti püsivara haldava teenuse OneCLI
rm -rf /opt/lenovo/onecli # Eemaldame vana versiooni
mkdir -p /opt/lenovo/onecli
cd /opt/lenovo/dss/dss-g-4.5a/opt/lenovo/dss/firmware/baseboard
tar xzf lnvgy_utl_lxce_onecli*tgz -C /opt/lenovo/onecli/
```

Sellega on Confluenti keskkond üles seatud. Edasi jätkatakse juhtserverite defineerimist, määratakse ära "*/etc/hosts*" failis juhtserverite võrgunimed ning nende IP aadressid

```
192.168.41.71 sol1 sol1.hpc.ut.ee
192.168.41.72 sol2 sol2.hpc.ut.ee
192.168.41.51 sol1-imm sol1-imm.hpc.ut.ee
192.168.41.52 sol2-imm sol2-imm.hpc.ut.ee
```

Defineeritakse Confluentis juhtserverite jaoks grupi ning määrame ära milliste hostinimedega Confluent pääseb ligi juhtserveritele haldusliidestele.

```
nodegroupdefine dssg bmc='{node}-imm' net.ib.hostname='{node}-ib0'
```

Järgmisena kirjeldatakse Confluent tarkvarale protokollid üle mille juhtservereid kontrollida ning sisestatakse vastavad kredentsiaalid haldusliidesesse sisse autentimiseks.

```
nodegroupattrib dssg deployment.useinsecureprotocols=firmware \
console.method=ipmi console.logging=full net.ipv4_gateway=192.168.41.50 \
ntp.servers=192.168.41.50 dns.servers=192.168.41.50 dns.domain=cluster

nodegroupattrib dssg -p bmcuser bmcpass crypted.rootpassword

Enter value for bmcuser:
Confirm value for bmcuser:
Enter value for bmcpass:
Confirm value for bmcpass:
Enter value for crypted.rootpassword:
Confirm value for crypted.rootpassword:
dssg: crypted.rootpassword: ********
dssg: secret.hardwaremanagementpassword: ********
dssg: secret.hardwaremanagementuser: ********
```

Edasi defineeritakse individuaalsed masinad mis kuuluvad gruppi dssg käsuga “*nodedefine sol1,sol2 groups=dssg*”. Pärast juhtserverite andmete sisestamist kontrollitakse varasemate seadistuste korrektsust

```
nodeattrib sol1,sol2 | collate
==================================
sol1,sol2
==================================
console.logging: full
console.method: ipmi
deployment.useinsecureprotocols: firmware
dns.domain: cluster
dns.servers: 192.168.41.50
groups: dssg,everything
hardwaremanagement.manager: sol1-imm,sol2-imm
net.ib.hostname: sol1-ib0
net.ipv4_gateway: 192.168.41.50
ntp.servers: 192.168.41.50
secret.hardwaremanagementpassword: ********
secret.hardwaremanagementuser: ********
```

Viimase kontrollkäsuna testitakse Confluenti võimekust kontrollida juhtservereid, selle kõige lihtsamaks viisiks on üle haldusliidese pärida kas serverid on sisse lülitatud või ei.

```
nodepower sol1,sol2 status
==================================
sol1,sol2 : offline
```

Pärast Confluendi seadistamist käib tarkvara paigaldamine juhtserveritele ühe käsuga "*dssg-install -N sol1,sol2*", kus siis sol1 ning sol2 on varasemalt defineeritud juhtserverite nimed. Tarkvara paigaldus võtab suurusjärgus 2h, mis on täisautomaatne ning selle jooksul tehakse kindlaks nii andmemassiivi kui ka juhtserverite konfiguratsioon ja paigaldatakse vastavad tarkvara ning püsivara versioonid ning seadistused. Pärast operatsioonisüsteemi ja DSS-G tarkvara paigaldust saab juhtserveritesse sisse logida üle SSH protokolli ning confluent keskkond ei ole enam vajalik.

3.2 Eeltöö failisüsteemi seadistamiseks

GPFS-i paigaldusele järgnev samm on riistvaralise seadistuse kontrollimine. Selleks paigaldab Lenovo kaasa mitu tööriista mille eesmärk on valideerida nii lahenduse topoloogia korrektsust kui ka individuaalsete kõvaketaste näitajaid. Lisaks on käesolevas etapis võimalik ka kettakandurite valideerimine ning nende firmware uuendamine. Siinkohal leidsime ka esialgse paigalduse juures tehtud vea tänu topoploogia vastavuse kontrollile ning kaabeldus sai korrektseks tehtud.

Riistvaralise seadistuse kontrollimiseks jooksutame käsku *dssgcktopology*.

```
Summary of the configuration found:
===================================
sol1,sol2
===================================
GNR enclosures found: J7435CN J7435CP J7435CR J7435CT J7435CV J7435CW
GNR server disk topology: DSS-G251 7Z73CTO LSI1BUS PCI 2,3,5,6,7 (match:
100/100)
GNR configuration: 6 enclosures, 26 SSDs, 0 empty slots, 444 disks total, 6
NVRAM partitions
```

```
Checking for number of servers...
Success: found 2 servers

Checking for enclosure issues...
Success: no enclosure issue detected

Checking for cabling issues...
Success: no cabling issue detected

Checking for drive issues...
Success: no drive issue detected

Checking for drive capacities...
Success: no discrepancy detected

Validating the product name and FRU of the drives...
Success: no unmatched drive detected
```

Lisaks kontrolliti ka individuaalsete kõvaketaste jõudlust käsuga "*dssgckdisks sol1,sol2*". DSS-G paigaldusjuhend soovitab välja vahetada andmekandjad mille jõudlus erineb märgatavalt ülejäänutest kõvaketastest [10]. Kõvaketaste jõudluses erisusi ei täheldatud ning riistvaraline konfiguratsioon loeti korrektseks.

Järgnevalt seadistati juhtserveritel erinevaid süsteemiteenuseid, nii määrati ajavöönd ning NTP teenus seadistati kasutama HPC valdusalas asuvaid kellaaega pakkuvaid servereid.

Edasi, seadistati juhtserverite võrguadapterid. Riigihanke ühe tehnilise tingimusena telliti juhtserveritega koos liiasusega kaks 100GbE võrguliidest mis tagavad juhtserverite kõrgkäideldavuse ning veakindluse riistvaralise vea puhul. Võrguliideste agregeerimiseks loodi kõigepealt uus virtuaalne võrguseade mis reklaamib vastava TCP kanali peal agregeerimise ohje protokolli LACP(Link Aggregation Control Protocol). Järgnevalt seadistati kahe füüsilise võrguliidese ülemliideseks värskelt loodud agregeeritud liides. Viimase sammuna määratakse juhtserverite võrguliidestele ka IP aadress.

```
nmcli con add type bond con-name bond0 ifname bond0 bond.options
"mode=802.3ad, miimon=100, lacp_rate=1"
nmcli conn add type ethernet slave-type bond ifname ens4f1np1 master bond0
nmcli conn add type ethernet slave-type bond ifname ens1f1np1 master bond0
nmcli conn modify bond0 ip4 172.16.20.31/24 gw4 172.16.20.1
```

Infiniband tüüpi liideste seadistamises suuri muudatusi ei tehtud, kuna enamus konfiguratsioonisamme on tarkvara paigaldusega juba tehtud.

Pärast võrguliideste seadistamist kontrolliti kahe juhtserveri omavaheline suhtlus üle TCP/IP protokolli ning testiti ka veakindlust ühendades ükshaaval võrgukaableid lahti, samal ajal veendudes võrguliikluse toimimises.

3.3 GPFS-i esialgsed seadistused

Järgmine samm GPFS-i paigaldamisel on kontrollstruktuuride loomine. Hajusa failisüsteemina sätestatati juhtserverid omavahel üheks klastriks, mille liikmeks olevad klientserverid failisüsteemile hiljem ligi pääsevad. Esialgu loodud klastrisse kuulub kaks masinat, milleks on mõlemad juhtserverid *sol1* ja *sol2*

Vastvalminud klastri seadistusi saame kontrollida käsuga *mmlscluster*

```
[root@sol1 ~]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:          sol.hpc.ut.ee
  GPFS cluster id:            13430862242101189526
  GPFS UID domain:            sol.hpc.ut.ee
  Remote shell command:       /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:            CCR

 Node  Daemon node name       IP address  Admin node name
Designation
--------------------------------------------------------------------------
-------
   1   sol1                   172.16.20.31   sol1
quorum-manager
   2   sol2                   172.16.20.32   sol2
quorum-manager
```

3.4 Esialgse failisüsteemi loomine

GNR tüüpi failisüsteemi peamine töömehhanism on kõikide juhtserveriga ühendatud kõvaketaste sidumine hajusaks kettamassiiviks (*declustered array*). Hajusa massiivi peamine ülesanne on failisüsteemi seadistamisel määratud veakindluse tagamine ning andmeplokkide jaotamine kogumis asuvate ketaste

vahel. Lisaks, et mõlemad juhtserverid saaksid töötada sünkroonis, jaotatakse massiiv kaheks võrdseks taastegrupiks (*recovery group*) kus kumbki juhtserver haldab enda grupis olevaid andmekandjaid. Juhtserveri vea tekkimisel võtab teine juhtserver taastegrupi haldamise üle ning failisüsteemi töö jätkub.

Vastavate struktuuride loomise ajal peab juhtserveritel ka GPFS-i haldav teenus *mmfsd* jooksma, ning see käivitati mõlemal serveril käsuga *mmstartup*. Pärast GPFS-i teenuse kontrollimist, valmistatakse ette GNR-i struktuurid käsuga *dssgmkstorage sol1,sol2* , mis samal ajal määrab ka vastavale juhtserverile hallatava veagruppi.

```
Configuring Storage Scale RAID using mmvdisk...
#
sol1: Building Block nc_sol1, Step 1
sol1: RUN: mmvdisk nodeclass create --node-class nc_sol1 -N sol1,sol2
sol1: mmvdisk: Node class 'nc_sol1' created.
sol1:
sol1: Building Block nc_sol1, Step 2
sol1: RUN: mmvdisk server list --node-class nc_sol1 --disk-topology
sol1:
sol1: Building Block nc_sol1, Step 3
sol1: RUN: mmvdisk server configure --node-class nc_sol1 --recycle all <<<
yes
sol1: mmvdisk: Recording pre-conversion cluster configuration in
/var/mmfs/tmp/mmvdisk.configure.before.20210812
sol1:
sol1: mmvdisk: This command will shutdown GPFS on multiple nodes at the
same time.
sol1: mmvdisk: It is possible to lose quorum and cluster availability.
sol1:
sol1: mmvdisk: Do you wish to continue (yes or no)?
sol1: mmvdisk: Checking resources for specified nodes.
sol1: mmvdisk: Node class 'nc_sol1' has a paired recovery group disk
topology.
sol1: mmvdisk: Using 'default.paired' RG configuration for topology
'DSS-G251 7Z73CTO HDD LSI1BUS PCI 2,3,5,6,8'.
sol1: mmvdisk: Setting configuration for node class 'nc_sol1'.
sol1: mmvdisk: Recording post-conversion cluster configuration in
/var/mmfs/tmp/mmvdisk.configure.after.20210812
sol1: mmvdisk: Node class 'nc_sol1' is now configured to be recovery group
servers.
sol1: mmvdisk: Restarting GPFS on the following nodes:
sol1: mmvdisk:
```

```
sol1:
sol1: Building Block nc_sol1, Step 4
sol1: RUN: mmvdisk recoverygroup create --recovery-group sol1,sol2
--node-class nc_sol1 --match 100 -v no
sol1: mmvdisk: Checking node class configuration.
sol1: mmvdisk: Checking daemon status on node 'sol1.hpc'.
sol1: mmvdisk: Checking daemon status on node 'sol2.hpc'.
sol1: mmvdisk: Node 'sol2.hpc' has a paired recovery group disk topology.
sol1: mmvdisk: Node 'sol1.hpc' has a paired recovery group disk topology.
sol1: mmvdisk: Analyzing disk topology for node 'sol1.hpc'.
sol1: mmvdisk: Analyzing disk topology for node 'sol2.hpc'.
sol1: mmvdisk: Creating recovery group 'sol1'.
sol1: mmvdisk: Creating recovery group 'sol2'.
sol1: mmvdisk: Formatting log vdisks for recovery groups.
sol1: mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGTIP
sol1: mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGTIP
sol1: mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGTIPBACKUP
sol1: mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGTIPBACKUP
sol1: mmvdisk: (mmcrvdisk) [I] Processing vdisk RG002LOGHOME
sol1: mmvdisk: (mmcrvdisk) [I] Processing vdisk RG001LOGHOME
sol1: mmvdisk: Created recovery groups 'sol1' and 'sol2'.
```

Järmine samm failisüsteemi sätestamises on failisüsteemi loomine. Esialgne failisüsteem sai loodud nii riistvara testimiseks kui ta lihtsamate vigade otsimiseks, mida ei leitud. Lisaks salvestati esialgse failisüsteemi ülesseadmise käsu *dssgmkfs* väljund, selleks ära märkida protseduur millega hilisema testimise ajal failisüsteemi parameetreid täpsemalt kontrollida. Käsk *dssgmkfs*, mis loob failisüsteemi struktuurid, kasutab GPFS-i haldamiseks mõeldud käske ning neid taasesitades võimalik ka “käsitsi” ehitada failisüsteemi struktuurid.

```
DSS-G 4.5a
Parsing options: --nsd 2 --mdcode 3WayReplication --dcode 8+2p \
--blocksize 16 --clients 150 --size 100 --mdDA DA2 --dataDA DA1 -- \
sol1,sol2
#
Checking server model...
Identified DSS-G server(s):
==================================
sol1,sol2
==================================
```

```
Lenovo ThinkSystem SR650 (DSS-G gen2)
#
Checking whether all nodes belong to the same Storage Scale cluster...
Checking whether the Storage Scale daemon is active...
#
Configuring a Storage Scale file system using mmvdisk...
#
File System: Step 5
Since this is hybrid, the -g parameter is set to 100% for mmvdisk
--set-size
sol1: RUN: mmvdisk vdiskset define --vdisk-set mvs1 --recovery-group
sol1,sol2 \
--code 3WayReplication --block-size 1m --set-size 100% --nsd-usage
metadataOnly \
--storage-pool system --declustered-array DA2
sol1: RUN: mmvdisk vdiskset define --vdisk-set dvs1 --recovery-group
sol1,sol2 \
--code 8+2p --block-size 16m --set-size 100% --nsd-usage dataOnly
--storage-pool \
data --declustered-array DA1
#
File System: Step 6
sol1: RUN: mmvdisk vdiskset create --vdisk-set mvs1,dvs1
#
File System: Step 7
sol1: RUN: mmvdisk filesystem create --file-system helios --vdisk-set
mvs1,dvs1 \
--mmcrfs -A no -n 32 -T /gpfs/helios
sol1: RUN: x=/tmp/policy.cfg.2020-03-31.051645.20642 ; echo "rule 'default'
set \
pool 'data'" > $x ; mmchpolicy
```

Esialgse failisüsteemi hääletamise lõpus sai GPFS-i klastrile lisatud ka kolmas masin, selleks et garanteerida kahe juhtserveri vaheline “viigimurdja”. Kuigi ei ole testimise ajal kolmas server vajalik, oli server *ares15* selleks varasemalt määratud ning server seoti GPFS-i klastriga.

4. Testimine ja tulemused

Järgnevas peatükis käsitletakse failisüsteemi testimise tööriistu ja metoodikat. Lisaks sellele antakse ülevaade testimisprotsessis, saavutatud tulemustest, kettamassiivi lõppkonfiguratsioonist ning ka ideid järgnevate failisüsteemide testimiseks.

4.1 Tööriist *gpfsperf*

Testimiseks kasutati GPFS-iga kaasa pakitud tööriista nimega *gpfsperf*. Tegu on IBMi poolt arendatud tööriistaga, millega saab hõlpsalt valideerida GPFS failisüsteemi jõudlust. Kuna *gpfsperf* on kirjutatud just GPFS-i testimiseks, ei kasuta programm mitte kerneli taseme süsteemikutsungeid IO operatsioonide sooritamiseks, vaid esitab *gpfsperf* oma plokitehingute sooritamiseks kutsungid otse GPFS-i kontrolliva *mmfsd* teenusele, eesmärgiga eemaldada profileerimiselt operatsioonisüsteemi vahekiht, et tööriist saaks võimalikult täpselt raporteerida just riistvarast sõltuvaid tulemusi.

4.2 Esialgne testimine ning gpfsperfi parameetrite häälestamine

Pärast esialgse failisüsteemi ülesseadmist ning riistvarale korrektsuse kinnitamist, jooksutati esimene komplekt profiilimisteste selleks, et kontrollida testmetoodikat. Ettevalmistatud pythoni skriptiga jooksutati *gpfsperf* tööriista laia valiku testploki suurustega, vahemikus 4KB kuni 16MB, eesmärgiga paika seada vajalik tuumade arv tööriistale, milleks kujunes 20.

Lisaks, selgus esimestest tulemustest kiiresti, et metoodikas on veel puudujääke kuna *gpfsperf* raporteeris kõikide testitud ploki suurustega kas identseid tulemusi või oli selgelt aru saada et tulemus on suurusjärgu kiirem kui kõikide kõvaketaste teoreetiline lagi. Lähemal uurimisel selgus, et originaalselt valitud testimisala suurus 1TB on konkreetsele lahendusele liiga väike ning terve testfail saaliti juhtserverite vahemällu, mille suurus kahe serveri peale on 2.34TB. Tõstes testimisala suuruse vastavalt 10TB peale ning pikendades ühe testi jooksutamise aeg 300 sekundi peale hakkas tööriist raporteerima tulemusi mis on reaalelule lähemal. *Gpfsperfi* käsureale sai ka lisatud parameeter “-fsync” mille ülesandeks on testi lõpus kõik saalitud andmed siluda kõvaketaste peale.

Järgmine voor teste sai jooksutatud metadata massiivi suuruse kontrollimiseks. Nimelt, failisüsteemi alloleva RAID struktuuri seob kaheks hajusaks massiiviks (*declustered array)* käsk *dssgmkstorage*, kuid failisüsteemi enda häälestamise ajal on võimalik täpsustada mitu protsenti massiivist määrata metaandmete kasutuseks. Ülejäänud SSD massiivi osa oli plaanis kasutada suurema kettamassiivi puhverdamiseks, seda tingimusel et massiivi poolitamine ei muuda oluliselt metaandmete ligipääsu kiirust.

Vastava konfiguratsiooni katsetamiseks loodi kõigepealt failisüsteem, määrates kogu SSD ketaste massiivi ala metaandmete jaoks. Pärast esimese vooru *gpfsperf* põhiste testite jooksutamist ning failisüsteemi tavapärase töö kontrollimist, jooksutati metaandmete päringute testimiseks IO500 [11] komplektis olevat tööriista nimega *mdtest*. Tööriist profiilib oma olemuselt nii failide kui kasutade loomise, ligipääsu ja kustutamise aegu ning konkreetseks testkomplektiks valiti kombinatsioon 512 000 failist ja kaustast.

Pärast esialgse konfiguratsiooni tulemuste salvestamist, kustutati failisüsteemi struktuurid ning loodi uus failisüsteem nii, et failisüsteemi metaandmete salvestusalaks määrati 10% originaalselt SSD põhisest massiivist. Seejärel korrati algset komplekti teste et kõigepealt kontrollida failisüsteemi tööd ning seejärel *mdtest* tööriistaga sama kombinatsiooni 512 000 failist ning kaustast. Tulemuste[3] võrdlemisel on küll näha erinevust tulemuste standardhälbest, kuid 10 iteratiivse testi mood jäi 3% ulatusse, mis kinnitas piisavalt et metaandmete ligipääsu kiirust mõjutab RAID struktuur mitte salvestusala suurus.

4.3 Testid failisüsteemi ploki suuruse mõju hindamiseks

Kettamassiivi testimiseks sai välja töötatud protseduur mida on varem ka HPCs kasutatud. Eeltööna valmistati ette kolmandas peatükis jooksutatud käsu *dssgmkfs* väljundi põhjal jada *mmvdisk* käsklusi[4] mis failisüsteemi struktuurid ette valmistavad. *Mmvdisk* käsklustega on võimalik failisüsteemi struktuuride parameetreid täpsemini kontrollida erinevalt ümbrisskriptist *dssgmkfs*.

1. Kogu lahenduse failisüsteemi struktuurid kustutatakse, et ette valmistada järgmist testitavat konfiguratsiooni. Alles jäeti failisüsteemi all olevad RAID struktuurid mis lõplikust konfiguratsioonist ei erine.
2. Ette valmistatud skriptidega ehitati uute parameetriga failisüsteem, mis lisaks veel valmistas ette 10TB suuruse faili mille seest testprogramm tulemusi mõõdab.
3. 30 minutiline paus, selleks failisüsteemi struktuuride puhvrid tühjeneksid ning testimine oleks võimalikult ühtlane.

[3] Tulemustega saab tutvuda lisades 1 ja 2
[4] Failisüsteemi loomise eest vastutava skripti kood asub lisas nr. 3

4. Alustab testimise eest vastutav skript[5], mis jooksutab eri parameetritega testrakendust *gpfsperf* ning salvestab tulemused.

Kogu protseduuri juhtimist kontrollis veel lisaks kirjutatud skript[6], mille ülesandeks oli võtta eelnevalt määratud parameetrid testidele ning siis vastavalt käivitada alamskripti. Vastav juhtprogramm jooksutas *gpfsperf* tööriista siis nelja erinevat andmeploki suurusega 2Mb, 4Mb, 8Mb, 16Mb ning kolme erineva metaandme ploki suurusega 512Kb,1Mb,2Mb, kokku kombineerides siis 12 erinevat konfiguratsiooni failisüsteemist.

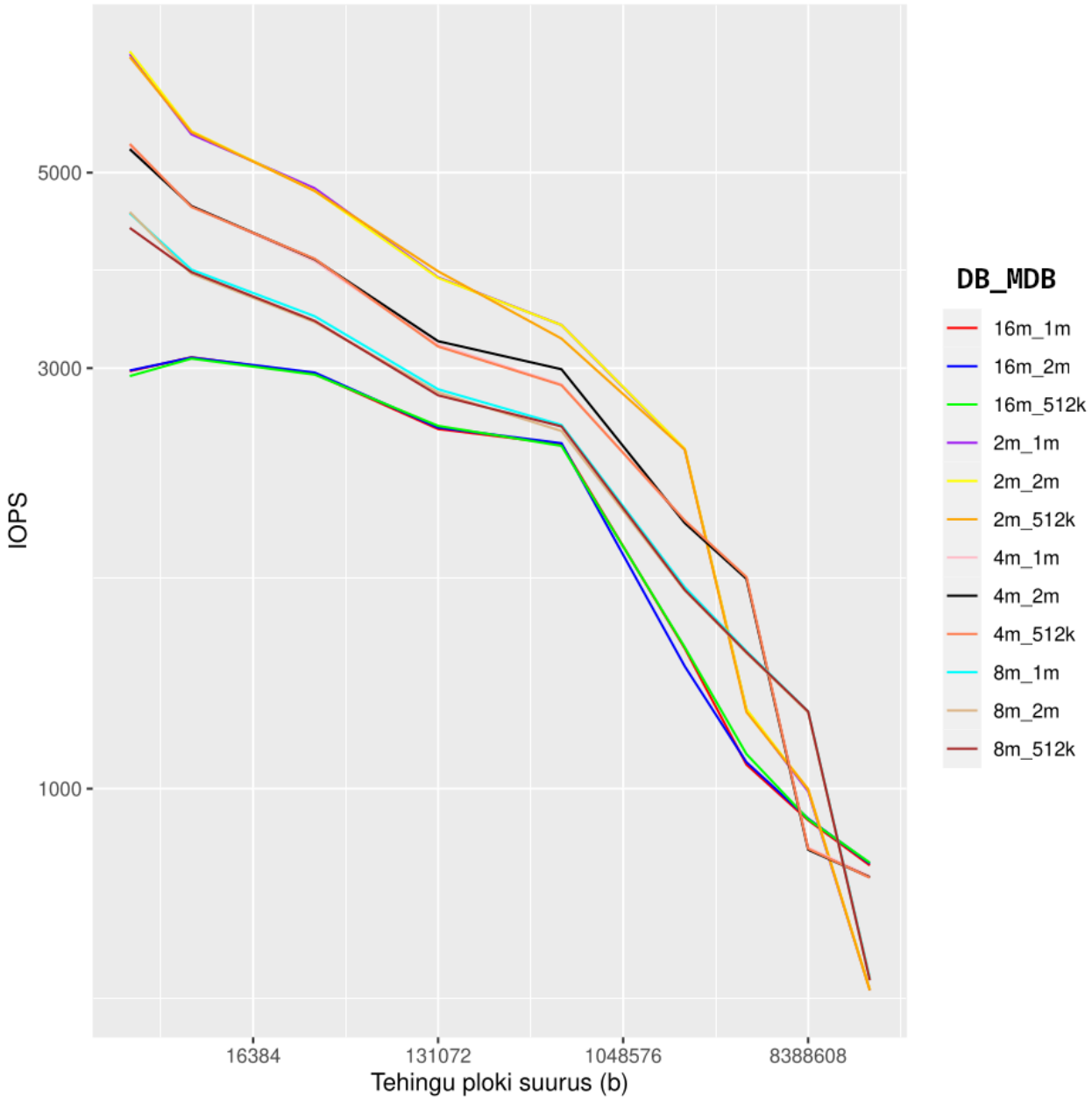
Igal erineval failisüsteemi versioonil seadistati *gpfsperf* kontrollima tehingu ploki suurusi järgnevatel väärtusel - 4Kb, 8Kb, 32Kb, 128Kb, 512Kb, 2Mb, 4Mb, 8Mb ja 16Mb. Määratud sai ka testmustride ja ligipääsu mustrite skeem, jättes välja ühesuunalised lugemis- ja kirjutamistestid ning kasutades ainult “mixrw” tüüpi profiili mis loeb ja kirjutab andmeid testimise ajal võrdselt. Testmustrit kontrolliti nii järjestikuse kui ka juhuslikuna.

4.4 Tulemused

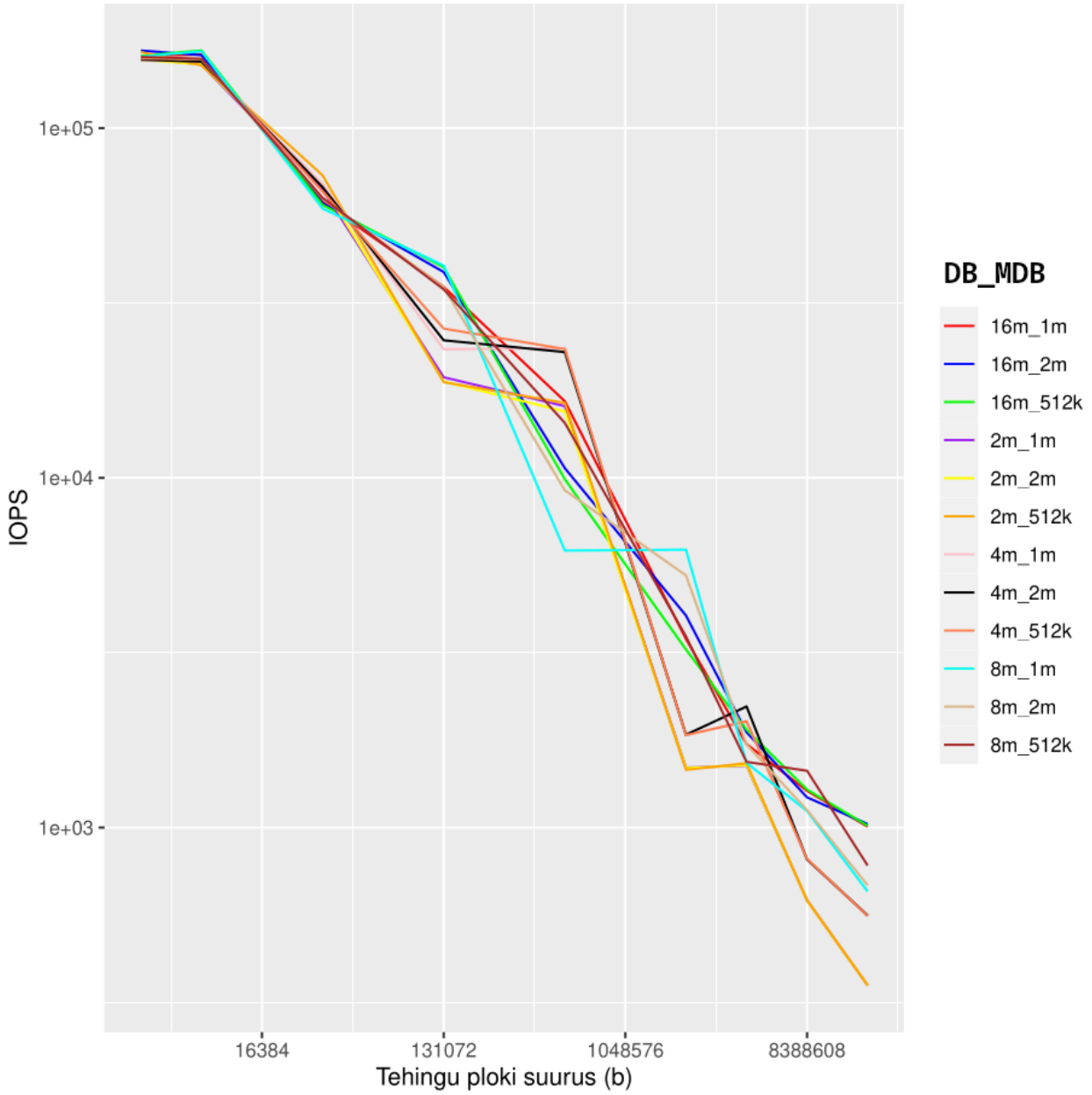
Testkomplekti jooksutamise tulemusena loodi 12 eri faili, milles leiduvad andmed koondati kokku neljaks eri graafikuks, vastavalt testmustrile ning võrreldes graafikult kuvatava tulemusena sisend-väljund operatsioonide arvu sekundis või andmeedastuskiirust.

Jooniste 3 ja 4 põhjal näeme, et kuigi järjestatud testides on IOPS eri ploki suurustega sarnane olenemate failisüsteemi konfiguratsioonist, on juhusliku päringumustriga eraldumas neli sarnase profiiliga konfiguratsiooni. Jooned graafikul koonduvad failisüsteemi andmeploki suuruse järgi ning metaandmete ploki suurus on palju vähem määravam failisüsteemi IOPSi jõudlusele. Lisaks, näeme võrdelist langust testploki suuruse kasvades mis viitab sellele, et failisüsteem peab rohkem riistvaralisi päringuid koondama ühe päringu alla.

[5] Skripti kood lisas nr. 5
[6] Skripti kood lisas nr. 4

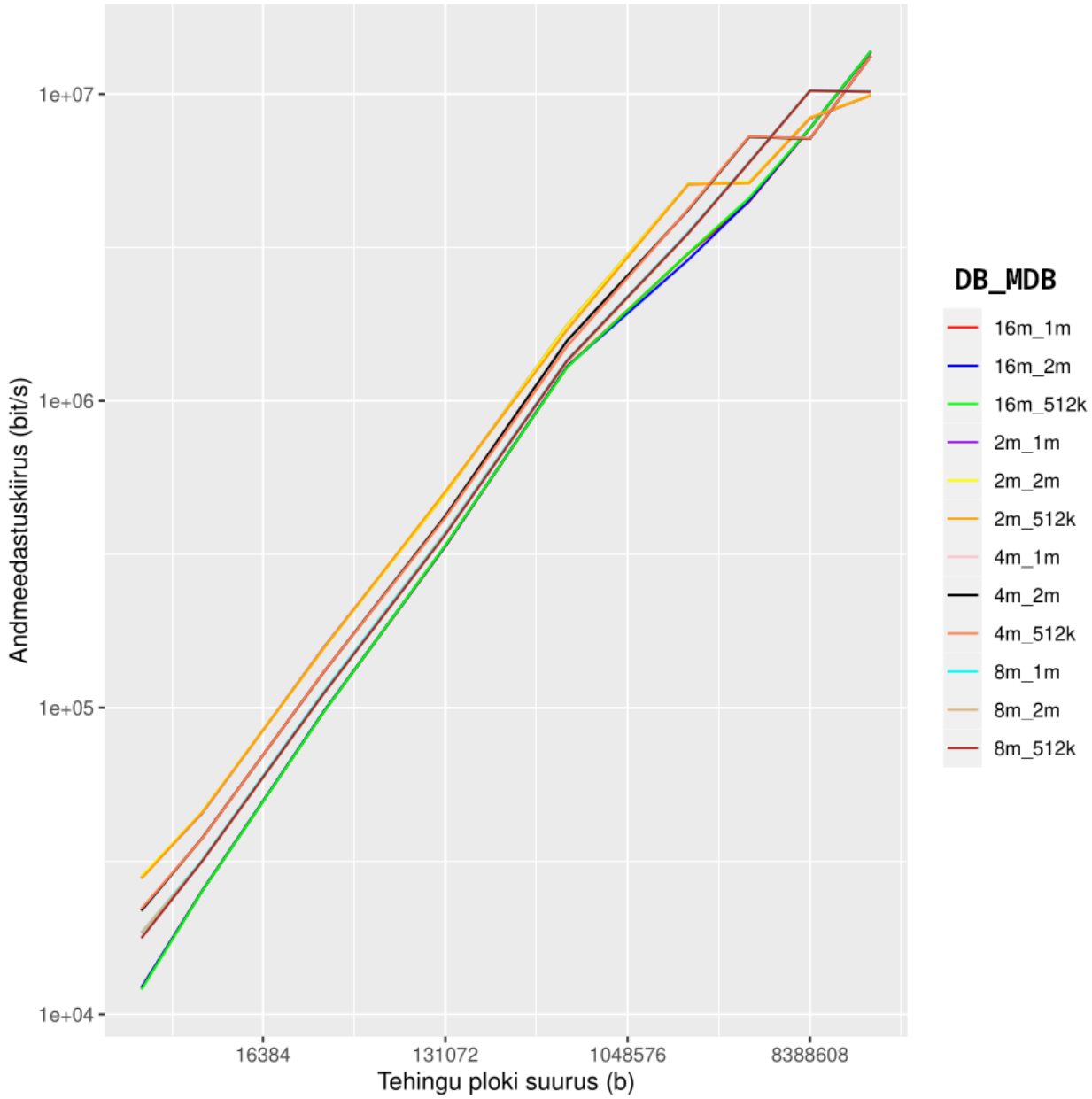


Joonis 3. Juhusliku testmustriga avaldunud operatsioonide hulk sekundis

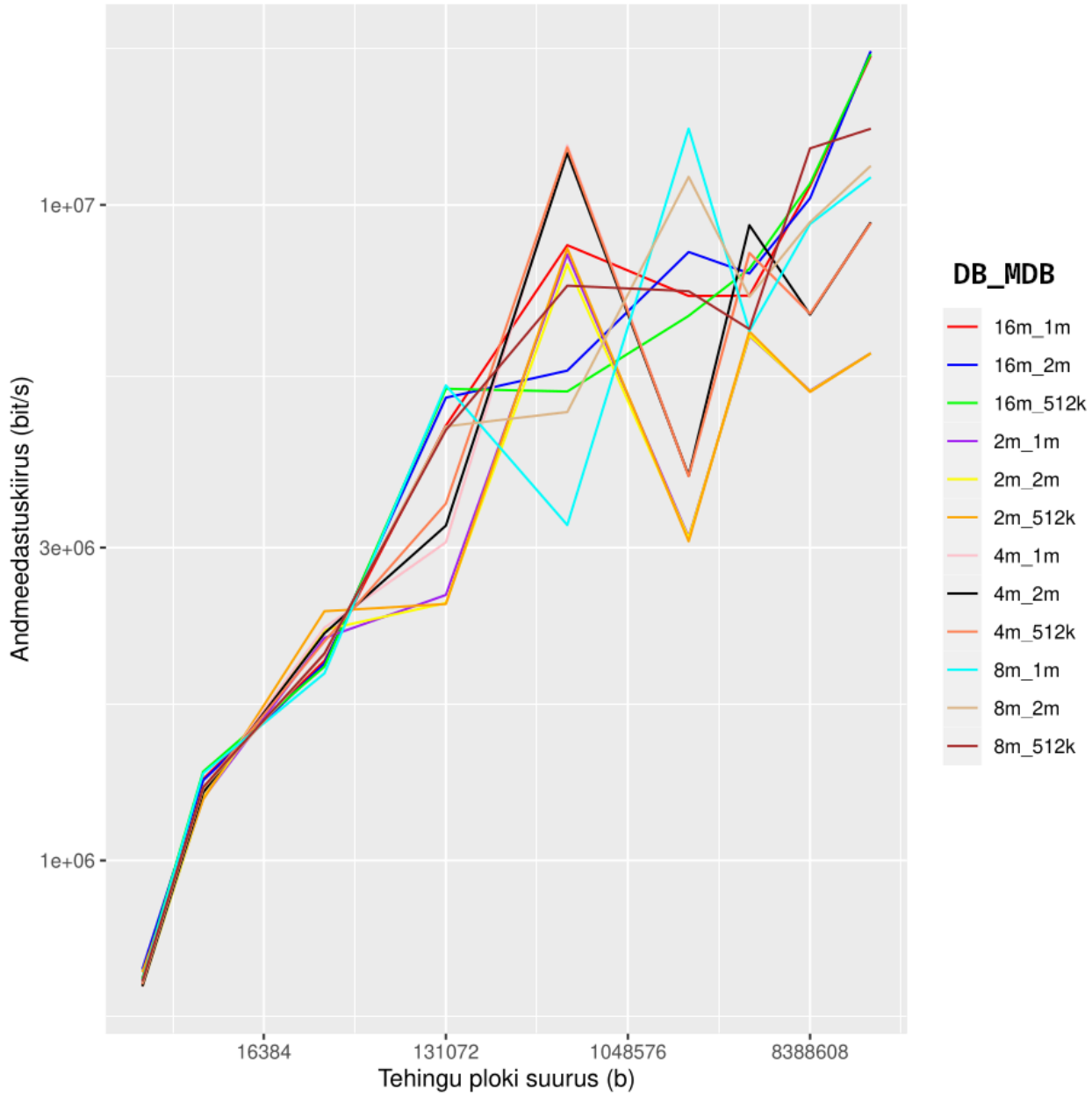


Joonis 4. Järjestikuse testmustriga avaldunud operatsioonide hulk sekundis

Vaadeldes andmekiiruse joonist 5, on kohati üllatav et juhuvalimiga testides on tulemused ühtlasemad ning joonisel 6 kuvatud järjestatud lugemis-kirjutamistesti tulemustes on suur varieeruvus. Siinkohal on hajutatud tulemustest joonisel 6 ka raske eraldada konkreetset konfiguratsioonis mis teistest parema tulemuse annab.



Joonis 5. Juhusliku testmustriga avaldunud andmeedastuskiirus



Joonis 6. Järjestikuse testmustriga avaldunud andmeedastuskiirus

Nelja graafiku peamine järeldus on suunav ikkagi tavapärasele loogikale - mida väiksem on failisüsteemi plokk seda rohkem tööd peab tegema failisüsteem päringu rahuldamiseks. IOPS-i väljendava joonise 3 kiireim tulemus koondub väikema 2M failisüsteemi ploki ümber, kuid andmeedastuskiiruses jääb sama konfiguratsioon suurusjärgu võrra alla 4M päringu suurusega testides. Lisaks ühtlustuvad suurtema päringutega juures ka 16M suuruse failisüsteemi ploki variatsioonid, sest failisüsteem peab esitama allolevale riistvarale vähem päringuid.

4.5 Failisüsteemi lõppkonfiguratsioon

Testkomplekti tulemuste analüüsimise lõpptulemusena määrati ka failisüsteemi lõplikud ploki suurused. Otsus langetati peamiselt andmeploki suuruse järgi, milleks määrati 8MB, kuna tulemustest on näha et 8MB suurune andmeplokk tagab piisava andmeedastuskiiruse nii järjestikku kui ja juhusliku lugemise-kirjutamise juures. Metaandmete ploki suuruse määramisel osutus vähem määravaks jõudlus, kuna andmetes ei väljendunud olulist erisust tulemustes.

Peatükis 4.2 mainitud *mdtest* tulemustest tingituna langetati ka otsus eraldada failisüsteemi metaandmete salvestusalaks 10% kogu SSD võimekusest, jättes ülejääva 90% hilisemaks seadistamiseks failisüsteemi puhvriks. Failisüsteemi *helios* lõplikuks salvestusmahuks sai seeläbi 5980TiB. Pärast edukat testimist ning HPC juhatusega konsulteerimist, sätestati failisüsteem oma lõplikus konfiguratsioonis ning algatati uuele andmemassiivile andmete pealelaadimist. Käesoleva lõputöö kirjutamise hetkeks on failisüsteem saavutanud 28% täituvuse HPC kasutajate kodukataloogide hoiustamisega (Joonis 7). Lõppseadistuste määramise kuni töö kirjutamise hetkeni failisüsteemi töös vigu ei ole täheldatud ning hiljem seadistatud SSD põhine puhver on kiirendanud osade töövoogude jooksutamist.

```
[root@sol1 ~]# mmdf helios --block-size=auto
disk                disk size  failure holds    holds                free                free
name                            group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 31.21 TB)
RG001VS001             3.805T        1 yes      no           2.953T ( 78%)        611.4M ( 0%)
RG002VS001             3.805T        2 yes      no           2.953T ( 78%)        622.6M ( 0%)
                -------------                         -------------------- -------------------
(pool total)           7.611T                               5.906T ( 78%)        1.205G ( 0%)

Disks in storage pool: data (Maximum disk size allowed is 23.32 PB)
RG001VS003             2.869P        1 no       yes          2.109P ( 74%)        3.751T ( 0%)
RG002VS003             2.869P        2 no       yes          2.109P ( 74%)         3.75T ( 0%)
                -------------                         -------------------- -------------------
(pool total)           5.738P                               4.218P ( 74%)        7.501T ( 0%)

Disks in storage pool: ssd (Maximum disk size allowed is 513.00 TB)
RG001VS002             51.77T        1 no       yes          11.19T ( 22%)        512.6G ( 1%)
RG002VS002             51.77T        2 no       yes          11.19T ( 22%)        514.6G ( 1%)
                -------------                         -------------------- -------------------
(pool total)           103.5T                               22.38T ( 22%)        1.003T ( 1%)

                =============                         ==================== ===================
(data)                  5.84P                                4.24P ( 73%)        8.504T ( 0%)
(metadata)             7.611T                               5.906T ( 78%)        1.205G ( 0%)
                =============                         ==================== ===================
(total)                5.847P                               4.246P ( 73%)        8.505T ( 0%)

Inode Information
-----------------
Total number of used inodes in all Inode spaces:          384157707
Total number of free inodes in all Inode spaces:           15506421
Total number of allocated inodes in all Inode spaces:     399664128
Total of Maximum number of inodes in all Inode spaces:   1986860032
```

Joonis 7. Väljund GPFS-i kettatäituvust kuvavast käsust *mmdf*

4.6 Võimalused järgmiste testide paremaks tegemiseks

Käesolevad tulemused näitavad küll selget trendi failisüstemi andmeploki suuruse ning IOPS-i vahel ning kinnitavad varasemat HPC kogemust DSS-G kettalahenduse jooksutamisel, kuid siiski on tulevikus võimalus arendada vastavat testkomplekti. Esimese täiendusena võib *gpfsperf* põhise testkomplekti kõrvale ka kaaluda mõne muu failisüsteemi jõudlustesti jooksutamist. Kuigi tööriista nimega *mdtest* kasutati SSD põhise salvestusala suuruse testimiseks, saab sama tööriistaga sügavamalt uurida metaandmete vastu sooritatud päringute jõudlust.

Mdtest-i kuulub ka IO500 testkomplekti, mida on laialt kasutatud erinevate HPC failisüsteemide jõudluse testimiseks. IO500 komplekt sisaldab endas ka IOR nimelise tööriista andmeedastuskiiruse teste mille tulemust saaks tulevikus võrrelda *gpfsperf* põhiste testidega.

Kokkuvõte

Tartu Ülikooli Teadusarvutuste keskuse kasvavate andmemahtude juures on oluline uuendada ja häälestada andmehoidlaid, selleks et tagada parem teenuse kvaliteet ning garanteerida andmemassiivi jõudlus HPC töövoogude jooksutamiseks.

Failisüsteemi ülesseadmiseks oli vajalik ette valmistada nii tarkvara paigaldamise keskkond, kui ka seadistada vajalikud võrguadapterid ning muud teenused, mis integreeruvad olemasoleva riistvaraga, mis on Teadusarvutuste keskusel juba olemas. Järgnenud testimine, mille ülesanne ei ole ainult andmemassiivi jõudluse hindamine vaid ka samas veakindluse testimine ning esialgsete vigade silumine on ainult osa ühe failisüsteemi elueast.

Pärast failisüsteemi esmast ülesseadmist ning olemasoleva taristuga sidumist, jooksutati kokku kaheteist erineva seadistusega andmemassiivil testkomplekt, määramaks failisüsteemi jõudlus nii andmeedastuskiiruses kui ka sisend-väljund operatsioonide arv sekundis. Tekkinud andmete analüüsimisel leiti Teadusarvutuste keskuse töövoogudega sobituv andmeploki ja metaandmete ploki suurus eesmärgiga garanteerida failisüsteemi piisav jõudlus erinevate töövoogude jooksutamisel.

Käesoleva töö tulemuseks on töö esitamise hetkel (august 2024) HPC süsteemidega täielikult integreeritud failisüsteem /gpfs/helios, kogumahutavusega 5980T ning 28% täituvusega. Failisüsteem *helios*e ülesandeks on vähemalt järgmise viie aasta jooksul talletada HPC klastri kasutajate kodukatalooge ning kindlasti lisandub talle tööülesandeid.

Viidatud kirjandus

[1] IBM. Storage Scale. https://www.ibm.com/products/storage-scale (02.08.2024)

[2] Haskin R. L. Tiger shark: a scalable file system for multimedia. *IBM Journal of Research and Development*, 1998, nr 42, lk 185-197

[3] IBM Documentation, Introducing IBM Storage Scale RAID. https://www.ibm.com/docs/en/storage-scale-ece/5.1.8?topic=administration-introducing-storage-scale-raid (02.08.2024)

[4] Perry T. IBM Spectrum Scale Concepts and features. *International Supercomputing Conference,* 2019, lk 32-33 https://www.spectrumscaleug.org/wp-content/uploads/2019/11/SC19-IBM-Spectrum-Scale-Concepts-and-features.pdf (03.08.2024)

[5] Lenovo. Lenovo Distributed Storage Solution for IBM® Storage Scale™ (DSS-G) https://support.lenovo.com/us/en/solutions/ht510974-lenovo-distributed-storage-solution-for-ibm-spectrum-scale-dss-g (02.08.2024)

[6] Riigihangete register. https://riigihanked.riik.ee

[7] IBM Documentation, mmcrfs command. https://www.ibm.com/docs/en/storage-scale/5.1.8?topic=reference-mmcrfs-command (10.08.2024)

[8] Shimpi A. L. The SSD Anthology. AnandTech, 2009, https://www.anandtech.com/show/2738 (02.08.2024)

[9]Lenovo Documentation. Confluent quickstart https://hpc.lenovo.com/users/documentation/confluentquickstart_el8.html (02.08.2024)

[10] Lenovo. DSS-G v4.5a Installation & Integration Guide. 2023

[11] IO500. https://io500.org (03.08.2024)

Lisad

I. Tööriista *mdtest* tulemused

Metaandmete salvestusala 10% mahutavusega

```
   Operation                   Max         Min         Mean        Std Dev
   ---------                   ---         ---         ----        -------
   Directory creation    :     24026.618   13380.233   22042.020   3075.237
   Directory stat        :     20432.644   19357.010   19913.506   312.344
   Directory removal     :     12247.488   11585.171   11870.159   186.516
   File creation         :     19337.055   11928.642   16547.417   2577.557
   File stat             :     28316.134   26827.383   27322.420   397.248
   File read             :     20519.319   19147.371   19579.929   421.442
   File removal          :     12460.840   9246.499    11308.229   1303.706
   Tree creation         :     480.852     187.810     334.988     117.739
   Tree removal          :     5.007       3.396       3.804       0.437
```

Metaandmete salvestusala 100% mahutavusega

```
   Operation                   Max         Min         Mean        Std Dev
   ---------                         ---         ---         ----
-------
   Directory creation    :     23618.919   14796.307   20288.493   3591.840
   Directory stat        :     20190.804   18730.339   19449.585   509.512
   Directory removal     :     11860.339   8917.055    11021.784   1063.274
   File creation         :     11658.583   8672.698    10765.171   772.823
   File stat             :     28013.754   24918.071   26752.063   967.875
   File read             :     19586.301   12929.847   17653.144   2314.097
   File removal          :     12280.570   11400.016   11805.381   326.567
   Tree creation         :     767.300     210.873     431.118     163.313
   Tree removal          :     5.407       3.260       3.861       0.545
```

II. Failisüsteemi struktuuri loomiseks ja kustutamiseks loodud skript vdisk_prepare_hybrid.sh

```
#!/bin/bash

while [[ $# -gt 0 ]]
do
key="$1"
case $key in
      -h|--help)
      help="true"
      shift
      ;;
      -c|--create)
      create="true"
      fs="$2"
      shift
      shift
      ;;
      -d|--delete)
      delete="true"
      fs="${2}"
      shift
      shift
      ;;
      -mbs|--metadata-block-size)
      mbs="${2}"
      shift
      shift
      ;;
      -dbs|--data-block-size)
      dbs="${2}"
      shift
      shift
      ;;
esac
sleep 1
done

help(){
```

```
    echo "$0 --delete  my_first_filesystem"
    echo "$0 --create my_first_filesystem --data-block-size XY[k|m]
--metadata-block-size XY[k|m]"
}

create(){
    #SOL - DA2 ssd, DA2 hdd
    mmvdisk vdiskset define --vdisk-set da1mvs1 --recovery-group sol1,sol2
--code 3WayReplication  --block-size "${mbs}" \
       --set-size 10% --nsd-usage metadataOnly --storage-pool system
--declustered-array DA1

    mmvdisk vdiskset define --vdisk-set da1dvs1 --recovery-group sol1,sol2
--code 8+2p  --block-size "${dbs}" \
       --set-size 90% --nsd-usage dataOnly --storage-pool ssd
--declustered-array DA1
    mmvdisk vdiskset define --vdisk-set da2dvs1 --recovery-group sol1,sol2
--code 8+2p --block-size "${dbs}" \
       --set-size 100% --nsd-usage dataOnly --storage-pool data
--declustered-array DA2

    mmvdisk vdiskset create --vdisk-set da1mvs1,da1dvs1,da2dvs1

    mmvdisk filesystem create --file-system "${fs}" --vdisk-set
da1mvs1,da1dvs1,da2dvs1 --mmcrfs -A no -n 256 -T /gpfs/"${fs}" \
       --write-cache-threshold 65536 -Q yes \
       --perfileset-quota --filesetdf --inode-limit 1149085440
    x=/tmp/policy.cfg.2021-10-26.201419.25570 ; echo "rule 'default' set
pool 'data'" > $x ; mmchpolicy "${fs}" $x
    mmlspolicy "${fs}"

    mmlsfs ${fs}  > fs_${fs}_${dbs}_${mbs}
    mmmount "${fs}" -a; sleep 10
    #dd if=/dev/zero of=/gpfs/"${fs}"/perf_test_file bs=10M count=102400
status=progress
    #gpfsperf makes the file faster than dd, but then sleep 30 is required
    gpfsperf create rand -n 10000g -th 10 -r 2m -shm -noinv -v -millis
10000 /gpfs/helios/perf_test_file
    sleep 30m


}
```

```
delete(){
    lsof /gpfs/"${fs}" | awk '!/COM/ {print $2}' | while read l;do kill -9
$l;done
    mmumount "${fs}" -a; sleep 5
    mmdelfs /dev/${fs}
    mmvdisk vdiskset delete --vdisk-set all
    for i in da1mvs1 da1dvs1 da2dvs1 ;do  mmvdisk vdiskset undefine
--vdisk-set $i --confirm; done
}

if [ -n "${help}" ] ; then help; fi
if [ -n "${delete}" ] && [ -n "${create}" ] ;then { echo no; exit 1; }; fi
if [ -n "${delete}" ] && [ -n "${fs}" ] ; then delete; fi
if [ -n "${create}" ] && [ -n "${fs}" ] && [ -n "${mbs}" ] && [ -n "${dbs}"
]; then create; fi
```

III. Failisüsteemi loomise ning testkomplekti jooksutamise eest vastutav skript main.py

```
#!/usr/bin/env python3

import subprocess
import sys
import logging
import perf

logging.basicConfig(filename='gpfs_perftesting.log', filemode='a',
level=logging.INFO)


DATA_BS_LIST = ['2m','4m', '8m', '16m']
SYSTEM_BS_LIST = ['512k','1m', '2m']
#Valid block sizes for RAID code 3WayReplication are 256k, 512k, 1m and 2m.

def delete_fs():
    logging.info('\n\n\nDeleting filesystem \n\n\n')
    proc = subprocess.Popen(['/root/commands/vdisk_prepare_hybrid_full.sh',
```

```
'--delete', 'helios'], stdout=subprocess.PIPE)
    std, err = proc.communicate()
    logging.info(std.decode())
    try:
       #succesful run has no err
       logging.info(err.decode())
    except:
       pass


def build_fs(data_bs, system_bs):
    logging.info(('\n\n\nBuilding filesystem with DATA blocksize {dbs} and
mETADATA(SYSTEm) blocksize {sbs}\n\n\n').format(dbs=data_bs,
sbs=system_bs))
    proc = subprocess.Popen(['/root/commands/vdisk_prepare_hybrid_full.sh',
'--create', 'helios', '--data-block-size', data_bs,
'--metadata-block-size', system_bs], stdout=subprocess.PIPE)
    std, err = proc.communicate()
    logging.info(std.decode())
    try:
       logging.info(err.decode())
    except:
       pass


def perftest_fs(fname):
    logging.info(perf.run(fname))

for dbs in DATA_BS_LIST:
    for sbs in SYSTEM_BS_LIST:
       print(dbs,sbs)
       delete_fs()
       build_fs(dbs,sbs)
       perftest_fs('perf_' + dbs + '_' + sbs + '.out' )
```

IV. *Gpfsperf* tööriista jooksutav ümbrisskript perf.py

```
#!/usr/bin/env python3

import sys
import time
import subprocess

PERF_BIN = '/usr/local/bin/gpfsperf'
TEST_BS = ['4k', '8k', '32k','128k','512k', '2M','4M','8M', '16M']
TEST_SIZE = '10000g'
TEST_THREADS = '20'
TEST_PATTERNS = ['seq', 'rand']
#TEST_STYLES = ['read', 'write', 'mixrw']
TEST_STYLES = ['mixrw']


def sizeof_blk(num, suffix='iB'):
      for unit in ['K','M','G','T','P','E','Z']:
      if abs(num) < 1024.0:
            return "%3.1f %s%s" % (num, unit, suffix)
      num /= 1024.0
      return "%.1f%s%s" % (num, 'Y', suffix)

def run(fname):
    std_output = ''
    test_results = []
    for pattern in TEST_PATTERNS:
       for bs in TEST_BS:
             for style in TEST_STYLES:
                   print('Starting perf with params:', style, pattern, bs)
                   out = subprocess.check_output([PERF_BIN, style, pattern,
'-fsync', '-r', bs, '-n', TEST_SIZE, '-th', TEST_THREADS, '-nolabels',
'-millis', str(300*1000),'/gpfs/helios/perf_test_file'])
                   test_results.append(out.decode()[:-1])


    #op pattern fn recordSize nBytes fileSize nProcs nThreads strideRecs
inv dio shm fsync cycle reltoken aio osync datarate oprate latency util'
    header = '{style:<10} {pattern:^10} {testsize:^10} {blocksize:^10}
```

```
      {datarate:<15} {oprate:<15} {latency:<9} {util:<8}   {b_transfer:<10}
'

    print(header.format(style='STYLE', pattern='PATTERN',
testsize='TEST_SIZE', blocksize='BLOCK_SIZE', datarate='DATARATE',
oprate='IOPS', latency='LATENCY', util='%UTIL',
b_transfer='BYTES_TRANSFERRED'))
    std_output += (header.format(style='STYLE', pattern='PATTERN',
testsize='TEST_SIZE', blocksize='BLOCK_SIZE', datarate='DATARATE',
oprate='IOPS', latency='LATENCY', util='%UTIL',
b_transfer='BYTES_TRANSFERRED')) + '\n'

    for result in test_results:
       spl = result.split()
       print(header.format(style=spl[0], pattern=spl[1],
testsize=TEST_SIZE, blocksize=sizeof_blk(int(float(spl[3])/1024)),
datarate=(sizeof_blk(int(float(spl[17]))) + '/s'), oprate=(spl[18] + '/s'),
latency=(spl[19] + 'ms'), util=spl[20],
b_transfer=sizeof_blk(int(float(spl[21])/1024))))
       std_output += (header.format(style=spl[0], pattern=spl[1],
testsize=TEST_SIZE, blocksize=sizeof_blk(int(float(spl[3])/1024)),
datarate=(sizeof_blk(int(float(spl[17]))) + '/s'), oprate=(spl[18] + '/s'),
latency=(spl[19] + 'ms'), util=spl[20],
b_transfer=sizeof_blk(int(float(spl[21])/1024))) + '\n'

    with open(fname, 'w') as f:
       for i in test_results:
             f.write(i + '\n')
       f.flush()

    return std_output

if(__name__ == '__main__'):
    run('manual_perf.out')
```

V. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Ott Eric Oopkaup,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose “Tartu Ülikooli Teadusarvutuste keskuse andmemassiivi ülesseadmine ja jõudluse testimine” mille juhendaja on Alo Peets, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Ott Eric Oopkaup
13.08.2024