ETRAN

Zbornik radova

62. Konferencija ETRAN 2018

&

5th International Conference IcETRAN 2018

Palić, 11-14.06.2018.

Društvo za ETRAN, Beograd & Akademska misao, Beograd

ETRAN Society, Belgrade & Academic Mind, Belgrade

Zbornik radova - 62. Konferencija za elektroniku, telekomunikacije, računarstvo, automatiku i nuklearnu tehniku, ETRAN 2018, Palić, 11 – 14. juna, 2018. godine

Proceedings of Papers – 5th International Conference on Electrical, Electronic and Computing Engineering, IcETRAN 2018, Palić, Serbia, June 11 – 14, 2018

Glavni urednik / Main Editor **Dejan Popović**

Urednici / Editors **Vladimir Katić, Nikola Jorgovanović**

Izdavači / Published by **Društvo za ETRAN, Beograd i Akademska misao, Beograd / ETRAN Society, Belgrade, Academic Mind, Belgrade**

Izrada / Production **Akademska misao, Beograd / Academic Mind, Belgrade**

Mesto i godina izdanja/ Place and year of publication **Beograd, 2018./ Belgrade, 2018**

Tiraž/ Circulation **300 primeraka / 300 copies**

ISBN 978-86-7466-752-1

www.etran.rs

CIP - Каталогизација у публикацији - Народна библиотека Србије, Београд

621.3(082)(0.034.2)
534(082)
004(082)(0.034.2)
681.5(082)(0.034.2)
621.039(082)(0.034.2)
66.017(082)(0.034.2)
57+61(048)(0.034.2)
006.91(082)(0.034.2)

ДРУШТВО за електронику, телекомуникације, рачунарство, аутоматику и нуклеарну технику. Конференција (62 ; 2018 ; Палић) ETRAN [Elektronski izvor] : zbornik radova / 62. Konferencija ETRAN2018, Kladovo, 05-08. juna, 2017. godine & 5th International Conference IcETRAN 2018 Palić, 11-14.07.2018. ; urednici, editors Vladimir Katić, Nikola Jorgovanović]. - Beograd : Društvo za ETRAN : Akademska misao = Belgrade : ETRAN Society : Academic Mind, 2018 (Beograd : Akademska misao). - 1 elektronski optički disk (CD-ROM) ; 12 cm Sistemski zahtevi: Nisu navedeni. - Nasl. sa nasl. ekrana. - Tekst ćir. i lat. - Radovi na srp. i engl. jeziku. - Tiraž 300. - Bibliografija uz svaki rad. - Abstracts.

ISBN 978-86-7466-752-1 (AM)

1. International Conference on Electrical, Electronic and Computing Engineering (5 ; 2018 ; Palić)

a) Електротехника - Зборници b) Акустика - Зборници c) Рачунарска технологија - Зборници d) Системи аутоматског управљања - Зборници e) Нуклеарна техника - Зборници f) Технички материјали - Зборници g) Биомедицина - Зборници h) Метрологија - Зборници

COBISS.SR-ID 268605452

Zbornik radova - 62. Konferencija za elektroniku, telekomunikacije, računarstvo, automatiku i nuklearnu tehniku, ETRAN 2018, Palić, 11 – 14. juna, 2018. godine ISBN 978-86-7466-752-1

Poređenje performansi homogenog i heterogenog Hadoop klastera na primeru TeraSort algoritma

Strahinja Đurković, Vladimir Ćirić, Natalija Stojanović

***Apstrakt*—Popularnost *Apache Hadoop* okruženja je u poslednjih nekoliko godina značajno porasla. Razlog tome je dobra skalabilnost i otpornost na otkaze, što ga čini dobrim kandidatom za kreiranje klastera sa velikim brojem računara. Obično se u takvim klasterima novi čvorovi dodaju nakon nekog vremena, kada se za to ukaže potreba, pa se samim tim razlikuju po performansama od čvorova koji se već nalaze u klasteru. Zbog ovoga većina klastera sa velikim brojem čvorova ima čvorove koji se međusobno razlikuju po performansama. Cilj ovog rada je testiranje i poređenje performansi heterogenog i homogenog *Hadoop* klastera sa standardnim YARN (eng. *Yet Another Resource Negotiator*) sistemom za upravljanje resursima klastera. Testiranje će biti izvršeno *TeraSort* algoritmom nad skupovima podataka različitih veličina, od 100MB do 5GB. Biće pokazano da se neravnomernom raspodelom zadataka na čvorove klastera može postići značajno ubrzanje.**

***Ključne reči*—Distribuirana obrada; *Apache Hadoop*; Heterogeni sistemi.**

I. Uvod

Razvoj informacionih sistema uticao je na značajan porast količine podataka koje treba obraditi. Zbog fizičkih ograničenja jednoprocesorski sistemi ne mogu ispuniti novonastale zahteve za performansama u obradi podataka [1]. Rešenje ovog problema predstavljaju distribuirani računarski sistemi, klasteri, koji pružaju mogućnost paralelne obrade podataka. Danas postoji dosta različitih implementacija distribuiranih računarskih sistema, među kojima se u poslednje vreme ističe *Apache Hadoop* [2].

Hadoop je razvijen u Java programskom jeziku po ugledu na Google-ov *MapReduce* razvojni okvir [3]. Karakterišu ga distribuirani fajl sistem – HDFS (eng. *Hadoop Distributed File System*), *MapReduce* implementacija za paralelnu obradu podataka i komponenta koja upravlja svim resursima klastera – YARN. Ovaj sistem je razvijen sa ciljem izvršenja na široko dostupnom i relativno jeftinom hardveru (eng. *commodity hardware*), odatle proizilazi jedan od njegovih najjačih aduta – visoka otpornost na otkaze. Performanse aplikacija koje se izvršavaju na *Hadoop* klasteru zavise kako od načina implementacije aplikacije tako i od opterećenja klastera u vreme njenog izvršenja. Jedan od dodatnih faktora koji utiče na performanse aplikacija jesu i same komponente klastera. Komponente klastera su mrežni uređaji i računari. U zavisnosti od hardverskih karakteristika računara razlikujemo heterogene i homogene klastere. Heterogene klastere čine računari sastavljeni od različitih hardverskih komponenti, dok homogene klastere čine računari sa identičnim hardverskim komponentama.

Heterogeni klasteri se u praksi često javljaju. Čest je slučaj da se u klaster čije performanse postanu nedovoljne dodaju novi čvorovi kada se za to ukaže potreba, pa se samim tim novododati čvorovi razlikuju po performansama od čvorova koji se već nalaze u klasteru. Ukoliko se zadaci raspoređuju ravnomerno na čvorove heterogenog klastera, najsporiji čvorovi će definisati brzinu celog klastera. Neravnomernom raspodelom zadataka na čvorove klastera može se postići značajno ubrzanje.

Ova tema zaokuplja pažnju različitih autora duži niz godina i pre pojave *Hadoop*-a. Postoje različita rešenja za implementaciju heterogenih distribuiranih sistema. Na primer, autori su u [4] razmatrali MPICH-G2 biblioteku za *grid* sisteme i pokazali ubrzanje do 88% na heterogenom sistemu. U [5] je pokazana implementacija heterogenog sistema korišćenjem MPI komunikacionog modela i GPU akceleratora. Što se *Hadoop*-a tiče, ranije verzije nisu imale mogućnost neravnomernog raspoređivanja zadataka [6]. Danas postoje rešenja za *Hadoop* koja omogućavaju neravnomernu raspodelu zadataka [7,8].

U cilju pružanja boljeg uvida i pomoći u proceni poboljšanja performansi klastera pri dodavanju čvorova sa boljim performansama i neravnomernom raspodelom zadataka, u ovom radu biće prezentovani rezultati testiranja i poređenja performansi heterogenog i homogenog *Hadoop* klastera sa standardnim YARN menadžerom resursa. Testiranje će biti izvršeno *TeraSort* algoritmom nad skupovima podataka različitih veličina, od 100MB do 5GB. Biće pokazano da se neravnomernom raspodelom zadataka na čvorove *Hadoop* klastera može postići značajno ubrzanje.

U drugom poglavlju ovog rada će detaljnije biti objašnjen način funkcionisanja *Hadoop* okruženja i biće dat precizniji opis heterogenog i homogenog *Hadoop* klastera. U trećem poglavlju će biti prikazan *TeraSort* algoritam koji je korišćen za testiranje klastera. U četvrtom poglavlju će biti prikazani rezultati testiranja, dok će zaključak biti dat u petom poglavlju.

Strahinja Đurković – student Elektronskog fakulteta Univerziteta u Nišu, Aleksandra Medvedeva 14, 18000 Niš, Srbija (e-mail: strahinja.djurkovic@elfak.rs).

Vladimir Ćirić – Elektronski fakultet, Univerzitet u Nišu, Aleksandra Medvedeva 14, 18000 Niš, Srbija (e-mail: vladimir.ciric@elfak.ni.ac.rs).

Natalija Stojanović – Elektronski fakultet, Univerzitet u Nišu, Aleksandra Medvedeva 14, 18000 Niš, Srbija (e-mail: natalija.stojanovic@elfak.ni.ac.rs).

 ISBN 978-86-7466-752-1

II. HETEROGENI I HOMOGENI HADOOP KLASTERI

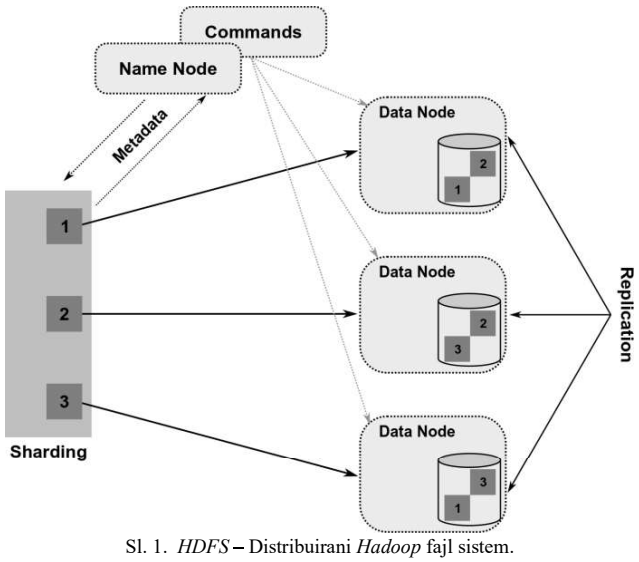
U ovom poglavlju će detaljnije biti opisane osnovne komponente na kojima se zasniva *Apache Hadoop*, kao i sami pojmovi heterogenih i homogenih klastera.

Kao što je rečeno u prethodnom poglavlju, *Hadoop* sistem sadrži tri ključne komponente: HDFS fajl sistem koji je zadužen za upravljanje podacima u okviru klastera, *MapReduce* implementaciju koja omogućava paralelno izvršavanje svih zadataka i YARN menadžer koji je zadužen za koordinisanje i upravljanje resursima u klasteru [2].

HDFS je razvijen po ugledu na *Google*-ov *GFS* distribuirani fajl sistem [3]. Arhitektura ovog rešenja je osmišljena kao *master-slave* arhitektura gde je glavni čvor nazvan *NameNode*, a čvor kojim on upravlja je nazvan *DataNode*. *Hadoop* klaster sadrži samo jedan *NameNode* koji radi u sprezi sa jednim *SecondaryNameNode*-om, kako bi omogućio brz oporavak klastera. U *NameNode*-u se nalaze svi *meta* podaci koji su vezani za klaster. Broj *DataNode*-ova je praktično neograničen, a može dostići i više desetina hiljada.

Na slici 1 je prikazana arhitektura HDFS-a [2]. Za upravljanje podacima se koristi *sharding* mehanizam. Ovaj mehanizam rukuje transferom podataka i njihovom replikacijom. Zaštita od otkaza i gubitka podataka je postignuta upravo ovim mehanizmom. Koeficijent replikacije je predefinisan na vrednost 3, ali je moguće izmeniti ga ili ga čak definisati na nivou pojedinačne datoteke. Datoteke se u HDFS-u obično dele na blokove veličine od 64MB do 128MB (moguće je definisati i blokove veličine 256MB ili 512MB).

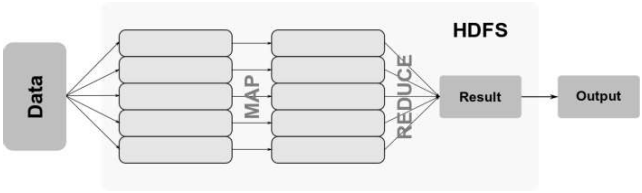
Svi *DataNode*-ovi redovno izveštavaju *NameNode* o stanju svih blokova paketom posebne strukture koji se simbolično naziva *Heartbeat.* Ovaj paket, pored informacija o aktivnosti, može nositi i informacije o blokovima podataka koji su sačuvani na tom *DataNode*-u. Ukoliko *NameNode* ne primi određen broj ovih paketa od nekog *DataNode*-a u definisanom vremenskom okviru taj *DataNode* se proglašava neaktivnim i kreće se u oporavak od otkaza koji se zasniva na *meta* podacima iz *NameNode*-a.



Sl. 1. *HDFS* – Distribuirani *Hadoop* fajl sistem.

Osnova *Hadoop MapReduce* implementacije počiva u *Google*-ovom *MapReduce* mehanizmu [3]. Ovaj mehanizam omogućava *Hadoop*-u paralelizaciju izvršenja zadataka i distribuiranje paralelizovanih delova svim čvorovima u klasteru. Izvršenje zadatka počinje deljenjem ulaznih podataka na blokove koji su predmet obrade *Map* dela programa. Rezultati ove faze izvršenja se klasifikuju i predstavljaju ulaz u drugu, *Reduce* fazu.

Na slici 2 je prikazan tok podataka kroz *MapReduce* mehanizam. HDFS izvršava prvi korak i reguliše distribuciju, smeštanje i replikaciju podataka. Nakon toga se, na svakom čvoru na kome se podaci nalaze, pokreće *Map* zadatak koji podatke transformiše u niz ključ-vrednost parova koji predstavljaju ulaz u sledeći deo procesa. Pre početka izvršenja *Reduce* zadataka svi ključ-vrednost parovi generisani u prethodnoj fazi se grupišu po vrednosti ključa i tako grupisani predstavljaju ulaz u *Reduce* fazu.



Sl. 2. *MapReduce* tok podataka.

Nakon izvršenja svih *Reduce* zadataka HDFS prikuplja sve generisane podatke i agregira ih u krajnji rezultat. Ovako generisan rezultat je podeljen na blokove, repliciran definisani broj puta i dostupan korisniku za čitanje.

Počev od verzije *Hadoop*-a 2.0, upravljanje fizičkim resursima i aplikacijama koje se izvršavaju na klasteru se prepušta YARN-u.

Na slici 3 je prikazana arhitektura YARN-a. Na glavnom čvoru se izvršava *ResourceManager* koji upravlja svim resursima u klasteru. *Scheduler* je deo *ResourceManager*-a zadužen za alokaciju resursa za sve aplikacije koje se izvršavaju na klasteru. U sprezi sa *ResourceManager*-om radi *NodeManager*. Na svakom računaru u klasteru se nalazi po jedna instanca *NodeManager*-a i ona nadgleda sve *Container*-e na računaru. *Container* predstavlja skup resursa neophodnih za izvršenje aplikacije. Na *Container*-u se izvršava poseban proces na *slave* čvoru u klasteru. *Container*-i su nadgledani od strane *NodeManager*-a. Uloga *ApplicationMaster*-a je da od *ResorceManager*-a zahteva resurse potrebne aplikaciji, kao i da sarađuje sa *NodeManager*-ima u izvršenju i praćenju progresa *MapReduce* zadataka aplikacije.

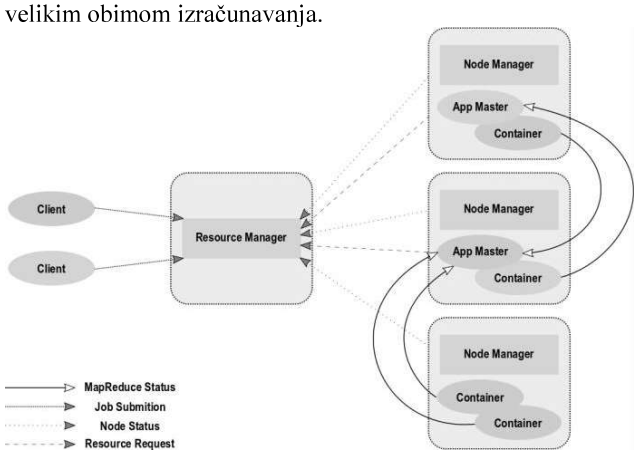
Hadoop je rešenje razvijeno za distribuirane računarske sisteme tj. klastere. Klastere možemo podeliti u dve grupe u zavisnosti od hardverske konfiguracije računara koji ga čine. Ukoliko su računari u klasteru potpuno identično konfigurisani radi se o homogenim klasterima. Računari u homogenom klasteru imaju procesore sa jednakim brojem jezgara i jednakim radnim frekvencijama, jednake količine sistemske memorije i diskove istog tipa sa jednakim kapacitetima. Ukoliko računari u klasteru nisu identični radi se o heterogenim klasterima. U praksi je vrlo čest slučaj heterogenih klastera. Iz praktičnih razloga nije uvek moguće u

 ISBN 978-86-7466-752-1

klaster nakon nekog vremena dodati novi čvor koji ima potpuno iste hardverske komponente kao postojeći čvorovi.

Pravilnom konfiguracijom *Hadoop* klastera moguće je staviti na znanje YARN komponenti, kao raspoređivaču zadataka, da čvorovi klastera nemaju jednake performanse, kako bi se to uzelo u obzir prilikom raspoređivanja zadataka. Ukoliko se ovo ne uradi, vreme izvršenja *Mapper*-a na najsporijem čvoru bi uslovilo kasnije pokretanje *Reducer*-a i eventualno usporilo celu aplikaciju.

Za testiranje različitih konfiguracija *Hadoop* klastera u ovom radu izabran je *TeraSort* algoritam, zbog mogućnosti da radi testiranja optereti klaster velikom količinom podataka i velikim obimom izračunavanja.



Sl. 3. *YARN* – Komponenta za upravljanje *MapReduce* mehanizmom.

III. TERASORT ALGORITAM I MAPIRANJE ALGORITMA NA HADOOP KLASTER

TeraSort algoritam se zbog svojih karakteristika često koristi za testiranje performansi *Hadoop* klastera [9,10]. Ovaj algoritam dolazi uz distribuciju *Hadoop* okruženja i podeljen je u tri podzadatka, koji su implementirani u *MapReduce* stilu: *TeraGen*, *TeraSort* i *TeraValidate*.

TeraGen predstavlja „slučajni“ generator podataka koji se kasnije sortiraju. *TeraGen* nasumično generiše parove ključ-vrednost da bi oni u kasnijim fazama bili sortirani po ključu. Ulazni parametri su željena veličina nasumično generisanih podataka i naziv direktorijuma u kome će podaci biti smešteni na HDFS-u. Ključ koji se generiše ovom prilikom je 10-to bajtna vrednost koja se kasnije koristi za sortiranje. Ova aplikacija predefinisano koristi dva *Map* zadatka za generisanje podataka za sortiranje i ne koristi nijedan *Reduce* zadatak. Broj *Map* zadataka je moguće postaviti prilikom pokretanja aplikacije.

Izlaz iz *TeraGen* aplikacije predstavlja osnovu za *TeraSort* aplikaciju. *TeraSort* kao ulaz očekuje direktorijum (na HDFS-u) u kome se nalaze podaci koje treba sortirati i direktorijum za smeštanje dobijenog rezultata.

Pre pokretanja samih *Map* zadataka vrši se raspoređivanje i grupisanje ključeva kreiranih u prethodnom koraku na pojedine mapere. Grupisani ključevi se distribuiraju svim *Reduce* zadacima, tako se osigurava da svaki *n*-ti *Reduce* zadatak dobija ključeve sa manjom vrednošću od ključeva koje je dobio *n*+1 *Reduce* zadatak, a većom od ključeva koje je dobio *Reduce* zadatak *n*-1. Nakon pokretanja aplikacije kreira se po jedan *Map* zadatak za svaki HDFS blok podataka generisanih *TeraGen* algoritmom. *Map* sortira sve podatke u okviru datog bloka na osnovu postojećih ključeva i emituje svoj izlaz. Predefinisana vrednost za broj *Reduce* zadataka je jedan, ali se može dodatnim ulaznim parametrom podešavati. Uloga *Reduce* zadataka je spajanje sortiranih podataka iz ključevima definisanog opsega u jednu datoteku. Broj delova rezultujuće datoteke na HDFS-u je jednak broju pokrenutih *Reduce* zadataka [9].

TeraValidate je treća aplikacija iz ovog skupa. Njen zadatak je provera izvršenja *TeraSort* algoritma. Ona prolazi kroz rezultujuću datoteku i proverava redosled sortiranih ključeva. U sledećem poglavlju će biti prikazani rezultati testiranja *Hadoop* klastera *TeraSort* algoritmom.

IV. REZULTATI TESTIRANJA

U cilju testiranja implementiran je jedan homogeni i jedan heterogeni *Hadoop* klaster. Homogeni klaster na kome je vršeno testiranje se sastoji od tri računara jednakih performansi. Svaki računar ima 2GB sistemske memorije, 20GB HDD prostora i po jedan virtuelni procesor. Heterogeni klaster umesto jednog od računara homogenog klastera sadrži jedan računar boljih performansi, koji ima 6GB sistemske memorije, 30GB NVMe SSD prostora i šest virtuelnih procesorskih jezgara. Svi računari su virtuelne mašine i izvršavaju se na računaru koji ima 16GB sistemske memorije, *AMD Ryzen* 5 1600 procesor, *Samsung 960 Evo Solid State* disk i običan HDD kapaciteta 500GB. *Hypervisor* korišćen za upravljanje virtuelnim mašinama je *VMware Workstation Pro 14*.

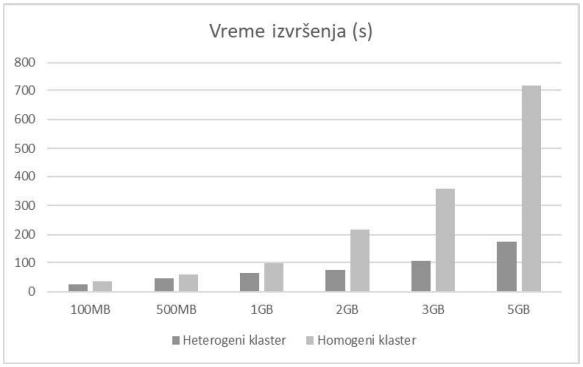
Hadoop verzija koja je korišćenja za implementaciju klastera je 2.8.3 sa standardnim YARN raspoređivačem zadataka. Za obe konfiguracije klastera zadate su minimalne i maksimalne vrednosti sistemske memorije po čvoru koja je stavljena na raspolaganje YARN-u. Ovi parametri nalaze se u konfiguracionim datotekama za HDFS i *MapReduce.* Na ovaj način *Hadoop* razlikuje heterogeni od homogenog klastera.

U cilju testiranja ovih klastera pokretani su *TeraGen* i *TeraSort* programi. *TeraValidate* zadatak nije razmatran. Testiranje je vršeno nad skupovima podataka veličine 100MB, 500MB, 1GB, 2GB, 3GB i 5GB. Prilikom pokretanja *TeraSort* algoritma korišćene su predefinisane vrednosti. Kako je predefinisana vrednost za broj *Reduce* zadataka jedan, to znači da je generisana samo jedna datoteka na HDFS-u veličine jednake veličini testiranog skupa. Ta datoteka se smešta na *DataNode*-u koji se nalazi na računaru na kome je izvršen *Reduce* zadatak. Na slici 4 su prikazana vremena izvršenja *TeraSort* algoritma na heterogenom i homogenom klasteru.

Sa slike se vidi da su vremena izvršenja *TeraSort* algoritma znatno kraća na heterogenom klasteru nego na homogenom klasteru. Ako se skupa sa ovim rezultatima posmatraju i rezultati prikazani na slici 5 može se zaključiti da YARN prilično favorizuje računar sa boljim performansama u heterogenom klasteru. Rezultati prikazani na ovim slikama predstavljaju prosečne izmerene vrednosti izvršenja TeraSort algoritma. Merenja su vršena po tri puta za svaki skup podataka i na heterogenom i na homogenom klasteru.

 ISBN 978-86-7466-752-1

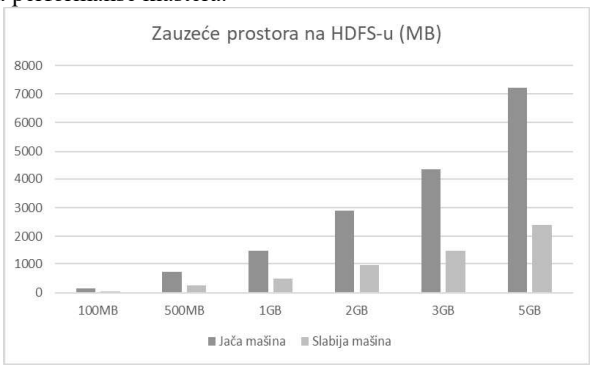
Na osnovu rezultata sa slike 4 može se zaključiti da se dodavanjem računara sa boljim performansama od postojećih računara u klasteru performanse klastera dodatno mogu poboljšati, a performanse samih računara bolje iskoristiti.



Sl. 4. Vremena izvršenja *TeraSort* algoritma u zavisnosti od veličine skupa podataka.

U primeru sa slike 4, YARN je prilikom raspoređivanja zadataka uvek izvršavao *Reducer* i smeštao datoteku na računar sa boljim performansama za slučaj heterogenog klastera (slika 5), iako su i računari sa slabijim performansama imali sasvim dovoljno resursa za ovu obradu. Ovo je značajno uticalo na skraćivanje ukupnog vremena izvršenja algoritma.

Potrebno je pomenuti i to da na performanse *Hadoop* klastera, pored hardverske konfiguracije, uticaj ima i komunikacija koja se odvija između komponenti. U zavisnosti od implementacije aplikacije koja se na klasteru izvršava može doći do prebacivanja velike količine podataka na računare sa boljim performansama, što može negativno uticati na performanse klastera.



Sl. 5. Zauzeće prostora na heterogenom klasteru nakon izvršenja *TeraSort* algoritma.

V. Zaključak

U cilju pružanja boljeg uvida i pomoći u proceni poboljšanja performansi klastera pri dodavanju čvorova sa boljim performansama i neravnomernom raspodelom zadataka, u ovom radu su prezentovani rezultati testiranja i poređenja performansi heterogenog i homogenog *Hadoop* klastera sa standardnim YARN raspoređivačem zadataka. Testiranje je izvršeno *TeraSort* algoritmom nad skupovima podataka različitih veličina, od 100MB do 5GB. Pokazano je da se neravnomernom raspodelom zadataka na čvorove klastera može postići značajno ubrzanje. Na osnovu rezultata merenja može se zaključiti da se dodavanjem računara sa boljim performansama u klaster mogu dodatno poboljšati performanse *Hadoop* aplikacija, a performanse samih računara bolje iskoristiti.

Zahvalnica

Ovaj rad je delimično podržan sredstvima sa projekta Ministarstva prosvete, nauke i tehnološkog razvoja br. TR32012.

Literatura

[1] Ćirić, Vladimir, Aleksandar Cvetković, and Ivan Milentijević. "Yield analysis of partial defect tolerant bit-plane array." Computers & mathematics with applications 59.1 (2010): 98-107.
[2] Lam, Chuck. Hadoop in action. Manning Publications Co., 2010.
[3] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: a flexible data processing tool." Communications of the ACM 53.1 (2010): 72-77.
[4] Allen, Gabrielle, et al. "Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus." Supercomputing, ACM/IEEE 2001 Conference. IEEE, 2001.
[5] Song, Fengguang, and Jack Dongarra. "A scalable framework for heterogeneous GPU-based clusters." Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures. ACM, 2012.
[6] Rao, B. Thirumala, et al. "Performance issues of heterogeneous hadoop clusters in cloud computing." arXiv preprint arXiv:1207.0894 (2012).
[7] *Xu, Xiaolong, Lingling Cao, and Xinheng Wang. "Adaptive task scheduling strategy based on dynamic workload adjustment for heterogeneous Hadoop clusters." IEEE Systems Journal 10.2 (2016): 471-482.*
[8] VishnuVardhan, Ch Bhaskar, and Pallav Kumar Baruah. "Improving the performance of heterogeneous Hadoop cluster." Parallel, Distributed and Grid Computing (PDGC), 2016 Fourth International Conference on. IEEE, 2016.
[9] O'Malley, Owen. "Terabyte sort on apache hadoop." Yahoo, available online at: http://sortbenchmark. org/Yahoo-Hadoop. pdf,(May) (2008): 1-3.
[10] Huang, Shengsheng, et al. "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis." Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on. IEEE, 2010.

Abstract

The popularity of Apache Hadoop environment in recent years has significantly increased. The reason for the increasing popularity is good scalability and resilience, which makes it a good candidate for creating a cluster with a large number of nodes. Usually in such clusters, new nodes are added after a while, when more processing power is needed. It is not uncommon that newly added cluster nodes differ in performance compared to the nodes that are already in the cluster, creating a heterogeneous structure. The goal of this paper is to evaluate and compare the performance of heterogeneous and homogeneous Hadoop clusters with standard YARN task scheduler. Evaluation will be performed using TeraSort algorithm with data sets from 100MB to 5GB.

Evaluation and comparison of homogeneous and heterogeneous Hadoop clusters using TeraSort algorithm

Strahinja Đurković, Vladimir Ćirić, Natalija Stojanović