Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jaka Plut

Vzpostavitev sistema Hadoop MapReduce v študijske namene

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Ljubljana 2016

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jaka Plut

Vzpostavitev sistema Hadoop MapReduce v študijske namene

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Matjaž Kukar

Ljubljana 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Raziščite možnosti za čimbolj enostavno inštalacijo in konfiguracijo (simuliranega) porazdeljenega sistema Hadoop MapReduce. Pomagajte si s prednameščenimi virtualkami, kot je na primer Cloudera, ki jih lahko na dovolj zmogljivem fizičnem računalniku namestite in konfigurirate za simulirano porazdeljeno izvajanje. Z razumljivimi primeri pojasnite in ilustrirajte programski model MapReduce. Vaše postopke preverite tako v simuliranem kot v dejanskem porazdeljenem okolju in ovrednotite njihovo skalabilnost na konkretnem primeru.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jaka Plut sem avtor diplomskega dela z naslovom:

Vzpostavitev sistema Hadoop MapReduce v študijske namene

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matjaža Kukarja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 9. februarja 2016 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Matjažu Kukarju za mentorstvo pri izdelavi diplomskega dela, družini, Tjaši in Jožetu za podporo tekom študija ter sošolcem in vsem ostalimi, ki so mi, vede ali nevede, kakorkoli pomagali na študijski poti.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CDH	Cloudera's distribution including Hadoop	Clouderina distribucija sistema Hadoop
DNS	Domain name system	sistem internetnih domenskih imen
DHCP	Dynamic Host Configuration Protocol	protokol za konfiguracijo dinamičnega gostitelja
YARN	Yet Another Resource Negotiator	MapReduce 2
KVM	Kernel-based Virtual Machine	Virtualizacijsko okolje za operacijski sistem Linux

Povzetek

Namestitev in konfiguracija porazdeljenega sistema Hadoop MapReduce je časovno precej zamudna in terja temeljito ter dosledno upoštevanje navodil. Kot takšna lahko povzroča precej nevšečnosti novim uporabnikom, ki bi se želeli spoznati s programskim modelom MapReduce.

Cilj diplomskega dela je raziskati možnosti za čim enostavnejšo namestitev in konfiguracijo porazdeljenega sistema Hadoop. Diplomsko delo se osredotoča na vzpostavitev porazdeljenega sistema Hadoop s pomočjo programske rešitve ameriškega podjetja Cloudera Inc., imenovane CDH, in temelji na platformi Apache Hadoop. Podjetje je izdalo več različic programske opreme CDH – trenutno je najnovejša CDH 5.5, ki jo je moč poganjati na različnih distribucijah operacijskega sistema Linux. V diplomskem delu so zapisani postopki in nasveti, ki so potrebni za uspešno vzpostavitev tovrstnega porazdeljenega sistema. V tekstu preučujemo različne možnosti namestitve in vzpostavitve sistema predvsem s pomočjo virtualizacije. Poleg omenjenega podrobno opisujemo in s primeri ilustriramo izvajanje poslov MapReduce. Na koncu izvedemo še kratko analizo, ki primerja učinkovitost (skalabilnost) izvedbe poslov MapReduce na enem, dveh in več vozliščih.

Ključne besede: Hadoop, namestitev, Linux, Cloudera.

Abstract

Installation and configuration of Hadoop MapReduce distributed system is quite time-consuming and requires thorough compliance with instructions. As such, it can cause considerable inconveniences to new users who would like to get familiar with the MapReduce programming model.

The aim of this thesis is to research the possibilities for straight forward installation and configuration of Hadoop distributed system. The thesis focuses on creating a distributed system using Hadoop software with the help of a solution called CDH, developed by an American company Cloudera Inc. The solution is based on Apache Hadoop platform. The company has released several versions of CDH software. The latest available CDH 5.5 can be run on different distributions of Linux operating system. The bachelor thesis is comprised out of instructions and tips which are necessary for the successful setup of such distributed system. The text researches various setup options and the creation of a system using predominantly virtualization. Furthermore, we are describing in detail, with examples, the process of running MapReduce jobs. In the end, we have a brief analysis which compares performance (scalability) of MapReduce jobs run on one, two and more nodes.

Keywords: Hadoop, setup, Linux, Cloudera.

Poglavje 1

Uvod

Živimo v času informacijske dobe, ko smo izpostavljeni neizmernim količinam podatkov, ki jih človeški um ni zmožen predelati in iz njih izluščiti dragocenih informacij. Prišli smo celo do točke, ko smo s pomočjo računalnikov ustvarili in zajeli tako velike količine podatkov, da jih z dano tehnologijo enostavno nismo zmogli analizirati in iz njih pridobiti informacij, saj je bilo podatkov enostavno preveč. V zadnjih letih se je za pojav, ko je tehnologija zmožna v kratkih časovnih intervalih pridelati petabajte podatkov, uveljavil termin "masovni podatki" ali s tujko, ki nam je bolj domača, "Big data". Poleg tega živimo v družbi in okolju, kjer pridobljene informacije v pravem trenutku lahko predstavljajo konkurenčno prednost in posledično poslovno priložnost.

V odgovor na problem, kako obvladati enormne količine podatkov in iz njih izluščiti informacije, ki prinašajo dodano vrednost, so strokovnjaki iz podjetja Google razvili programski model MapReduce, ki je zmožen predelati masovne količine podatkov in za svoje delo ne potrebuje drugega kot množice osebnih računalnikov [2].

Poleg rešitve, ki je bila razvita pod okriljem podjetja Google, obstajajo tudi druge implementacije programskega modela MapReduce – med njimi je tudi Hadoop MapReduce, ki je ena od glavnih komponent porazdeljenega sistema Hadoop. Vzpostavitev in konfiguracija porazdeljenega sistema Ha-

doop brez pomoči programske opreme, ki bi to naredila avtomatizirano, je dolgotrajen in relativno zahteven proces. V kolikor se hočemo spoznati s programskim modelom MapReduce in sistemom Hadoop, moramo najprej namestiti stabilno različico sistema Hadoop, ki jo lahko prenesemo s strežnikov podjetja Apache. Na vsako vozlišče, ki ga hočemo vključiti v gručo (ang. cluster), je potrebno prenesti, razširiti in pognati namestitvene datoteke. Ko namestimo vse potrebne sistemske pakete, je potrebno sistem še pravilno nastaviti. Potrebno je nastaviti okoljske spremenljivke, ki jih potrebujejo Hadoopovi demoni za svoje delovanje, prilagoditi programske skripte, nastaviti konfiguracijske parametre vozlišča, določiti mesto datotečnemu sistemu, kjer bodo dnevniški zapisi demonov, nastaviti širok nabor parametrov in podobno. Pri vsem naštetem je za upravljanje gruče potrebno poznati ukaze za upravljanje z vozlišči in njihovimi vlogami [3]. To so ukazi za zaganjanje in ustavljanje vlog na vozliščih, formatiranje vozlišč in ostalo. Celoten postopek zahteva kar nekaj časa in truda, zato v diplomskem delu predstavljamo alternativni način, ki občutno skrajša čas vzpostavitve in konfiguracije sistema ter je veliko preprostejši od opisanega.

V nadaljevanju so torej predstavljeni postopki, navodila in nasveti, kako vzpostaviti lasten porazdeljeni sistem, ki podpira izvajanje poslov MapReduce ter je preprostejši in krajši od klasične namestitve, opisane v prejšnjem odstavku. Navodila za vzpostavitev porazdeljenega sistema so sestavljena za najbolj popularno implementacijo z imenom Hadoop. Namenjena so uporabnikom, ki se prvič srečujejo s Hadoopom in želijo preizkusiti njegove zmožnosti ter vse, kar ponuja. Navodila pri tem upoštevajo, da povprečen uporabnik nima na voljo večje množice osebnih računalnikov, ki bi jih lahko povezal v gručo, zato so prilagojena v smeri, da lahko skoraj vsak, ki ima nekoliko boljši računalnik ali dva povprečna, s pomočjo virtualizacije simulira porazdeljeni sistem. Z istimi navodili je moč sestaviti tudi večjo gručo virtualnih naprav, ki se poganja na množici osebnih računalnikov. Prav tako lahko predstavljena navodila služijo kot pomoč pri vzpostavitvi porazdeljenega sistema na fizičnih napravah brez uporabe virtualizacije, saj je večina

korakov vzpostavitve enakih. Korake, ki se neposredno navezujejo na uporabo virtualizacije, bi bilo mogoče analogno rešiti na fizičnih napravah. V navodilih je opisana raba virtualne naprave s prednameščeno programsko opremo, gre za programski rešitvi podjetja Cloudera (Cloudera Manager in CDH), ki sta v diplomskem delu podrobneje predstavljeni. Slednji bi bilo v primeru vzpostavitve porazdeljenega sistema na fizičnih napravah potrebno namestiti ročno.

Diplomsko delo je razdeljeno na več vsebinskih sklopov. Uvodu sledi poglavje 2 s predstavitvijo programskega modela MapReduce in njegovega demonstracijskega primera izvedbe na podlagi štetja pojavitev besed v poljubnem besedilu. Znotraj tega poglavja je predstavljena tudi platforma Apache Hadoop 2 z njenima sestavnima deloma, Hadoop Distributed File System (HDFS) in MapReduce znotraj platforme Hadoop, poglavje pa vsebuje tudi kratko predstavitev podjetja Cloudera ter njenih programskih rešitev CDH in Cloudera Manager, ki sta služili kot osnova za vzpostavitev porazdeljenega sistema. Poglavje 3 je posvečeno sami vzpostavitvi sistema Hadoop, ki zajema pregled sistemskih zahtev in zahtev omrežnih nastavitev, navodila za vzpostavitev nadzornega in delovnega vozlišča. V poglavu 4 so predstavljena navodila za poganjanja poslov MapReduce glede na programski jezik, ki smo ga uporabili reševanju problema. Temu sledita poglavje 5, ki govori o pisanju poslov MapReduce, in poglavje 6, v katerem so predstavljeni poskusi, njihovi rezultati in kratka analiza. Diplomsko delo zaključi poglavje 7 s sklepnimi ugotovitvami in oceno praktične uporabnosti predlagane rešitve.

Poglavje 2

MapReduce

MapReduce je programski model, namenjen vzporednemu procesiranju in generiranju velikih množic podatkov [6]. Programski model je zasnovan tako, da se samodejno porazdeljuje in ga je moč vzporedno poganjati na velikem številu vozlišč – tudi več tisoč. Vozlišča lahko v tem primeru predstavljajo kar običajni, osebni računalniki. MapReduce v času svojega izvajanja poskrbi za vse podrobnosti v zvezi z razdeljevanjem vhodnih podatkov na manjše dele, ki se jih posreduje vsakemu od vozlišč, vrstnim redom izvajanja programskih ukazov na vozliščih, rokovanjem z napakami na vozliščih ali celo odpovedmi, komunikacijo med njimi in podobno. Slednje omogoča programerjem, ki se še nikoli niso srečali z vzporednimi in porazdeljenimi sistemi, da brez težav izkoriščajo danosti porazdeljenih sistemov [6].

2.1 Programski model

Kot je že bilo navedeno, programski model MapReduce služi procesiranju ogromne količine podatkov, pri čemer se poslužuje vzporednega procesiranja. Kot že njegovo ime pove, bistvo programskega modela predstavljata funkciji **map** in **reduce**. V računalništvu sta bili omenjeni funkciji predstavljeni veliko prej kot programskih model MapReduce, in sicer v okviru **funkcijskega programiranja**. V okviru funkcijskega programiranje se srečamo s pojmom

funkcije višjega reda. To so funkcije, ki jih je moč uporabiti kot vhodni parameter ali celo izhod neke druge funkcije [8]. Med drugim v funkcije višjega reda spadata tudi funkciji **map** in **reduce**.

2.1.1 Funkciji *map* in *reduce* v okviru funkcijskega programiranja

Iz perspektive funkcijskega programiranja je *map* funkcija višjega reda, ki aplicira dano funkcijo, pravimo ji *transformer*, na seznamu vhodnih elementov in vrne seznam rezultatov. Funkcija *transformer* je aplicirana na vsakega od elementov vhodnega seznama in vrača enega ali več novih elementov. Primer psevdokode funkcije *map*, ki pretvori velike črke v male, je naslednji:

```
map(toLower)"abcDEFG12!@#" --> "abcdefg12!@#"
```

Vloga funkcije **reduce** oz. **fold** v okviru funkcijskega programiranja predstavlja funkcijo višjega reda, ki obdela seznam vhodnih podatkov v določenem zaporedju ter vrne končno vrednost. Pri tem uporabi funkcijo *combiner*, ki jo aplicira na paru elementov seznama in vrne rezultat, ki je (lahko) nadalje uporabljen pri obdelavi preostalih elementov seznama. Primer psevdokode funkcije *reduce*, ki izračuna vsoto števil [9]:

```
reduce (+) [1..5] --> 15
```

2.1.2 Funkciji *map* in *reduce* v okviru programskega modela MapReduce

Osrednji del programskega modela MapReduce predstavljata funkciji *map* in *reduce*, vendar obstajajo določene razlike v primerjavi s funkcijama, ki sta v rabi v okviru funkcijskega programiranja. Medtem ko bistvenih razlik pri funkciji *map* ni, se implementacija funkcije *reduce* bistveno razlikuje. Funkcija *reduce* podobno, kot to poznamo iz funkcijskega programiranja, aplicira funkcijo *combiner* na vhodne podatke. Vendar se v primeru programskega modela MapReduce funkcija *combiner* aplicira na **več podmnožic** seznama

vhodnih podatkov. Tako izračuna **množico** končnih rezultatov, medtem ko se pri funkcijskem programiranju *combiner* aplicira na **celotnem seznamu** vhodnih podatkov in izračuna **en** končni rezultat [9].

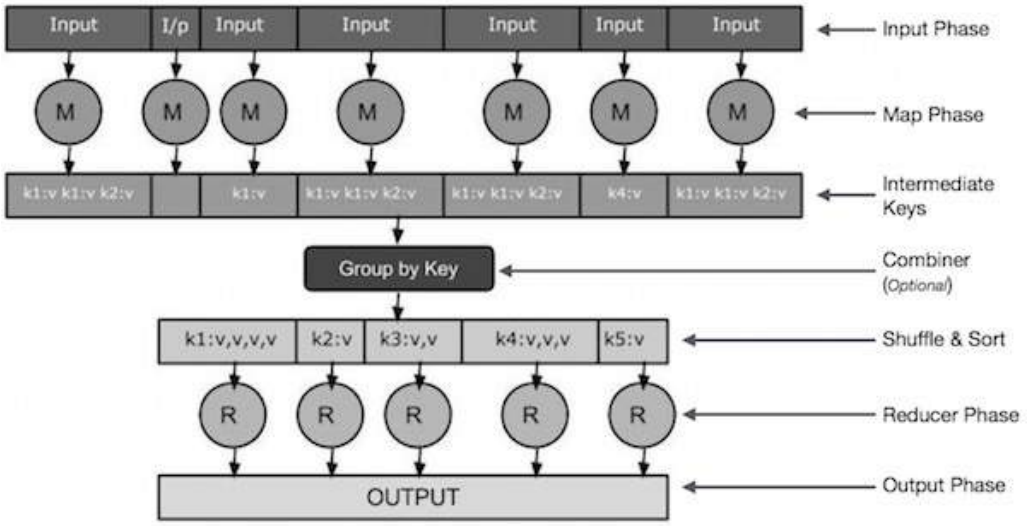
2.1.3 Pregled programskega modela MapReduce

Algoritem (slika 2.1), ki stoji za MapReduce, v grobem poteka po naslednjem vrstnem redu: v začetku posla so datoteke z vhodnimi podatki razdeljene na **bloke**. Število blokov hkrati predstavlja tudi število opravil **map**, ki se bodo izvedla v okviru posla. Programski model določi vozlišča, ki bodo nad bloki podatkov pognala funkcijo *map*. Na vsakem od vozlišč funkcija *map* vrne vmesne rezultate posla v obliki množice parov <*ključ, vrednost*> . Vsi vmesni rezultati so nato sortirani tako, da so pari z **enakim ključem** razvrščeni skupaj. Ko so vsi pari razvrščeni, so le-ti posredovani na vhod funkcije **reduce**, katere izhod gradi končni rezultat [9].

Dogodkom, ko se nad podatki izvaja funkcija map ali reduce, pravimo **opravilo map** oz. **opravilo reduce**. Prav tako kot opravila *map*, tudi opravila *reduce* tečejo porazdeljeno, vendar lahko njihovo število določi uporabnik sam. Razlog, da lahko opravila *reduce* tečejo porazdeljeno je ta, da se znotraj vsakega od opravil *reduce* za dan ključ obdelajo **vsi** pari s tovrstnim ključem. Posel MapReduce se zaključi, ko se zaključijo vsa opravila *map* in *reduce*. Rezultat posla je v obliki datotek, katerih število je enako številu *reduce* opravil [6].

V posebnih primerih, ko pride do velikega števila ponavljanj v vmesnih rezultatih, ki jih vrnejo opravila *map* in je funkcija *reduce* tako komutativna kot asociativna, lahko uporabnik specificira funkcijo *combiner*. Funkcija *combiner* običajno vsebuje enako logiko kot funkcija *reduce* in se jo uporablja za pohitritev posla, saj se izvede lokalno, na vsakem od vozlišč, ki so pognali funkcijo *map*. Funkcija *combiner* tako na vhod prejme vmesne rezultate, ki so produkt funkcije *map*, jih obdela in za razliko od funkcije *reduce*, ki vrne končni rezultat, vrne nov vmesni rezultat, ki se nato pošlje funkciji *reduce*. S tem, ko se že na vsakem od vozlišč reducirajo vmesni rezultati, se opravila

reduce izvedejo hitreje, kar posledično skrajša izvajanje celotnega posla [6]. Prav tako občutno razbremenijo omrežni promet med vozlišči.



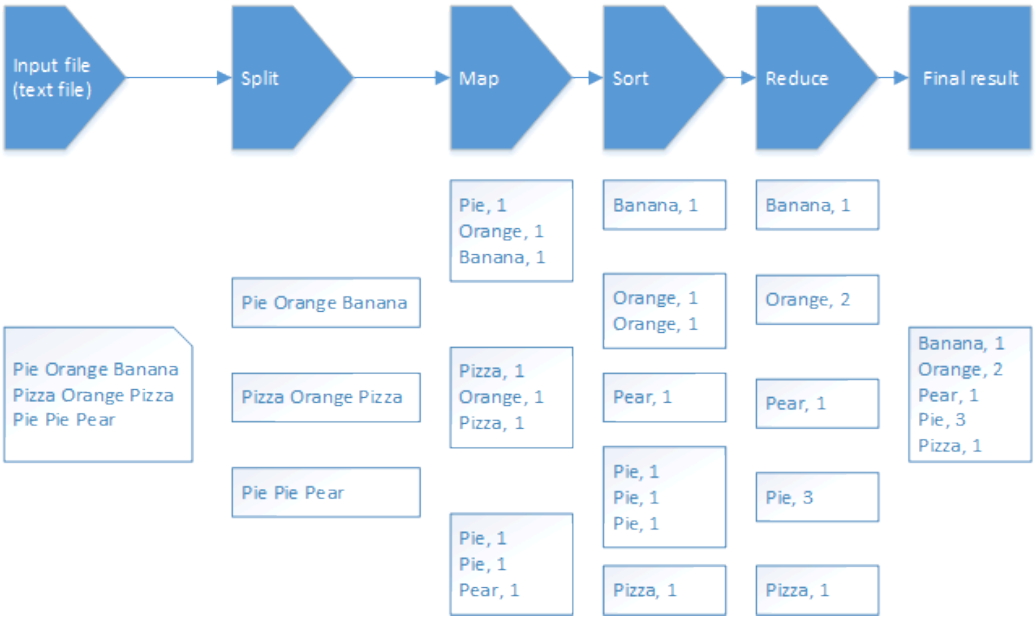
Slika 2.1: Visokonivojska predstavitev poteka posla MapReduce (`http://http://www.tutorialspoint.com/map_reduce/images/phases.jpg`).

2.2 Primer: štetje pojavitev besed v besedilu

Predstavljajmo si, da programskemu modelu MapReduce za vhod določimo poljubno besedilo v tekstovni datoteki (slika 2.2). V sklopu posla MapReduce je datoteka razdeljena na *bloke*. Vsak od blokov je poslan ločenemu vozlišču. Vozlišča na blokih poženejo funkcijo *map*, ki smo jo predhodno napisali in je specifična glede na naravo problema, ki ga rešujemo.

Vsako vozlišče požene funkcijo *map* nad blokom podatkov, ki se preslika v pare <*ključ, vrednost*> glede na algoritem v funkciji *map*. V našem primeru *ključe* predstavljajo posamezne besede besedila vhodne datoteke, *vrednost* pa število pojavitev besed oz. njihova frekvenca.

Funkcija *map* ustvari pare <*beseda, 1*> za vsako od besed z vhoda. Pri tem je pomembno poudariti, da se v funkciji *map* ne zgodi agregacija poja-



Slika 2.2: Potek izvajanja posla MapReduce na primeru štetja pojavitev besed (`http://www.widriksson.com/wp-content/uploads/2014/10/Hadoop-MapReduce-WordCount.png`).

vitev besed, tj. funkcija *map* ne izračuna števila pojavitev neke besede, saj je to naloga funkcije *reduce*.

Pari oz. vmesni rezultati, ki jih ustvari funkcija *map*, so nato sortirani. Razvrščeni so glede na ključ, kar pomeni, da bodo pari, ki imajo za ključ enako besedo, razvrščeni skupaj. Nato se nad vmesnimi rezultati požene funkcija *reduce*, ki glede na algoritem, spisan v funkciji, agregira zbrane podatke. V našem primeru gre skozi pare, agregira tiste z enakim ključem, sešteje njihove ponovitve in vrne par, kjer ključ predstavlja posamezno besedo, vrednost pa predstavlja število ponovitev te iste besede [5].

2.3 Apache™Hadoop 2

Apache™ Hadoop je odprtokodni projekt. Osredotoča se na razvoj programske opreme, ki omogoča porazdeljeno procesiranje velike množice podatkov na gručah, sestavljenih iz običajnih računalnikov [10]. Hadoop, po zgledu Googlovega sistema MapReduce, v splošnem delimo na dva dela. Prvi se ukvarja s hrambo podatkov, medtem ko drugi zagotavlja porazdeljeno procesiranje podatkov [17].

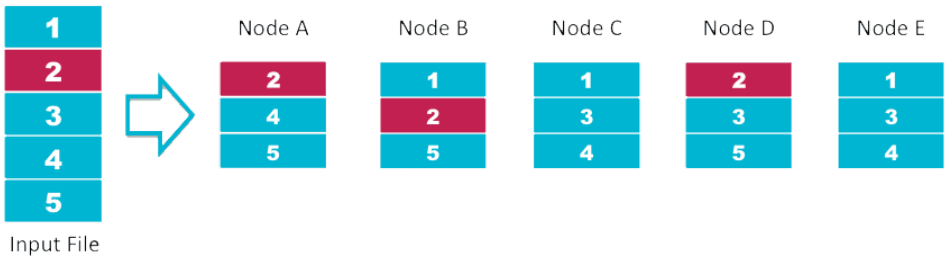
Hadoop je sestavljen iz štirih modulov, in sicer:

1. HDFS (Hadoop Distributed File System),
2. Hadoop MapReduce,
3. YARN (Yet Another Resource Negotiator) in
4. Hadoop Common [11].

2.3.1 HDFS

HDFS oz. Hadoop Distributed File System predstavlja porazdeljeno hrambo podatkov, ki se uporablja pri pisanju programov, ki tečejo na platformi Hadoop [12]. Gre za porazdeljeni datotečni sistem, ki skrbi, da je hramba podatkov skalabilna, odporna na odpovedi in poceni. Prav tako zazna napake in motnje, ki se pojavljajo na vozliščih, in jih po svojih najboljših močeh rešuje ter kompenzira, tudi v primerih, ko pride do napak na trdih diskih ali celo do odpovedi celotnega vozlišča [17].

HDFS hrani podatke na množici vozlišč, povezanih v gručo. Datoteke so razdeljene na **bloke**, pri čemer je vsak od blokov prenesen na več kot eno vozlišče. Tovrstno repliciranje blokov (slika 2.3) zagotavlja dostopnost podatkov v primeru odpovedi katerega od vozlišč in tako pripomore k boljšemu delovanju sistema, saj lahko druga vozlišča pridejo do podatkov na več različnih mestih, kar izboljša njihovo dostopnost [17].



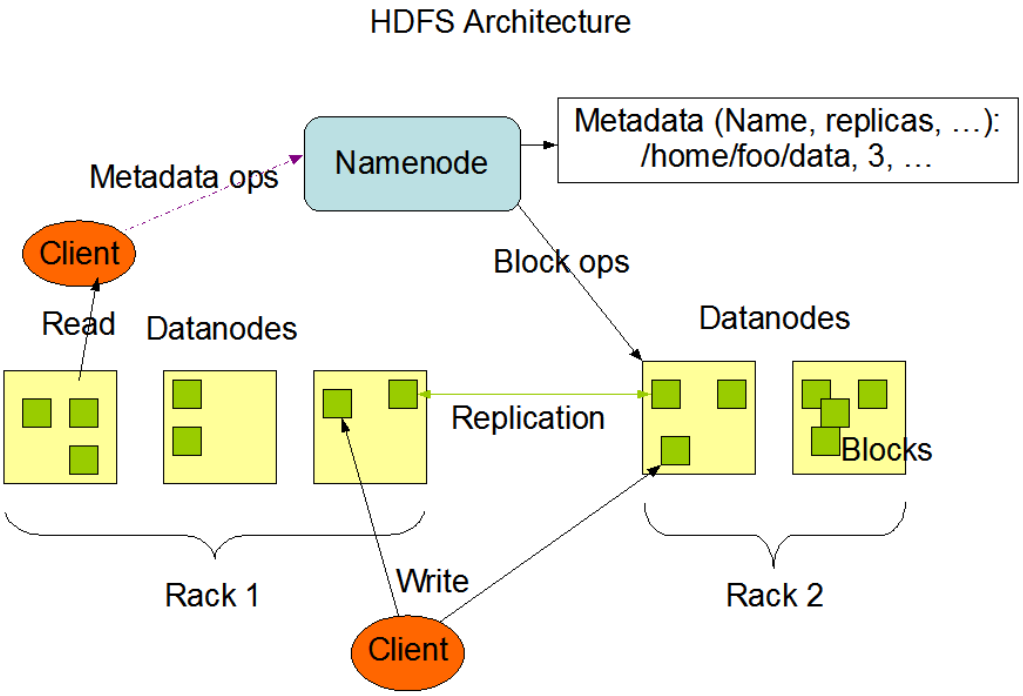
Slika 2.3: Shema ponazarja hrambo podatkov v HDFS. S številkami so označeni bloki, ki se hranijo na vozliščih gruče. Rdeča barva ponazarja replikacijo bloka na vozliščih (`http://www.cloudera.com/content/dam/cloudera/product-assets/hdfs-data-distribution.png`).

HDFS zagotavlja dostopnost podatkov prek kontinuiranega nadzora vozlišč, ki sestavljajo gručo. Pri tem HDFS vodi kontrolno vsoto za vsakega od shranjenih blokov. Ko HDFS pregleduje bloke, poskrbi, da se kontrolne vsote ujemajo. V kolikor se kontrolna vsota katerega od blokov ne ujema, ga sistem povrne v veljavno stanje tako, da na drugih vozliščih poišče blok, ki je v veljavnem stanju in z njim nadomesti blok, katerega kontrolna vsota se ne ujema. Prav zaradi te sposobnosti, tudi v primeru odpovedi vozlišč, HDFS še vedno dobro deluje, saj se kopije blokov nahajajo na ostalih vozliščih, ki jih s pridom uporablja. Vse to prispeva k dejstvu, da lahko zavoljo lastnosti in kvalitet, ki jih nudi Hadoop, za vozlišča uporabimo povsem običajne osebne računalnike [17].

HDFS sloni na arhitekturi *gospodar-suženj* (ang. master-slave). Gruča je sestavljena iz enega vozlišča, ki ima nameščeno vlogo *NameNode*, in več vozlišč z vlogo *DataNode*. Vozlišče z vlogo *NameNode* je glede na arhitekturo *gospodar*. To vozlišče ureja tako sistemski imenski prostor kot tudi dostop odjemalcev do datotek. Poleg vozlišča z vlogo *NameNode* so v sam proces vključena tudi vozlišča z vlogo *DataNode*. To so vozlišča, ki skrbijo za hrambo podatkov znotraj gruče. Običajno ima vsako vozlišče v gruči pripisano vlogo *DataNode*. Podatki, ki jih uporabnik preda v procesiranje platformi, so razdeljeni v enega ali več blokov, ki jih hranijo vozlišča z vlogo

DataNode [17].

Naloga vozlišča, ki ima nameščeno vlogo *NameNode*, je izvajanje operacij, kot so odpiranje, zapiranje datotek, njihovo preimenovanje in podobno. Med drugim skrbi tudi za preslikavo blokov na vozlišča z vlogo *DataNode* (slika 2.4), ki skrbijo za streženje zahtevkov po branju in pisanju v datoteke, ki hranijo podatke. Poleg tega izvajajo tudi kreacijo blokov, njihovo brisanje in replikacijo glede na zahtevke, ki pridejo z vozlišča z vlogo *NameNode* [13].

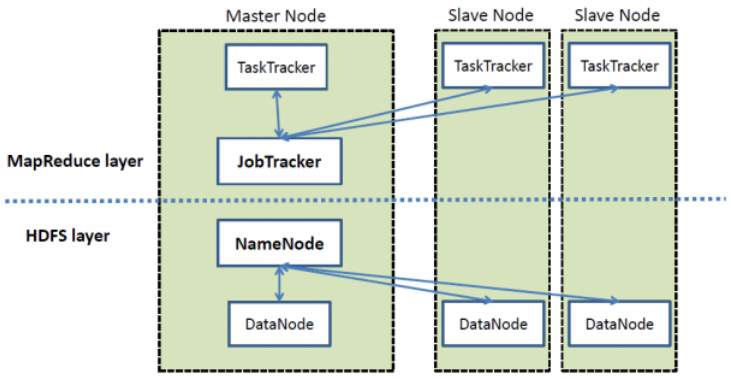


Slika 2.4: Ahitektura sistema HDFS.
(`https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html`).

2.3.2 Hadoop MapReduce

Osnovna ideja MapReduce je predstavljena v poglavju 2, na tem mestu pa povejmo še nekaj o njegovi arhitekturi znotraj Hadoopa. Programsko okolje Hadoop MapReduce je namenjeno pisanju aplikacij, ki obdelujejo velike količine podatkov (tudi več terabajtov). Procesu znotraj MapReduce pravimo *posel* (ang. *job*). Posel Hadoop sestavljata *opravilo map* (ang. *map task*) in *opravilo reduce* (ang. *reduce task*) [20]. Kot že imeni opravil namigujeta, se znotraj *opravila map* vrši funkcija *map* in znotraj *opravila reduce* funkcija *reduce*.

Programsko okolje Hadoop MapReduce v sodelovanju s HDFS (slika 2.5) sestavljata dve vlogi, in sicer *JobTracker* in *TaskTracker* [19]. Vloga *JobTracker* je nameščena na eno vozlišče v gruči, medtem ko je *TaskTracker* nameščena na vsako od vozlišč gruče. Tudi tukaj gre za arhitekturo *gospodar-suženj*, pri čemer je *gospodar* vozlišče z nameščeno vlogo *JobTracker* in *suženj* vozlišče z vlogo *TaskTracker*. *Gospodar* skrbi za časovno usklajevanje poslov in opravil na *sužnjih*, jih nadzira in pošlje v ponovno izvajanje v primeru napak, medtem ko *suženj* izvaja opravila po navodilih *gospodarja* [20].



Slika 2.5: Visokonivojska predstavitev arhitektur Hadoop MapReduce in HDFS.
(`http://opensource.com/sites/default/files/resize/images/life-uploads/hadoop-HighLevel_hadoop_architecture-640x460.png`).

2.3.3 Razporeditev vlog HDFS in Hadoop MapReduce v gruči

Kot smo že omenili, znotraj HDFS poznamo dve vlogi in sicer *NameNode*, ki predstavlja *gospodarja* in *DataNode*, ki predstavlja *sužnja*. Vlogo *NameNode* ima lahko nameščeno le eno vozlišče v gruči. Podobno arhitekturo ima tudi Hadoop MapReduce, kjer *gospodarja* predstavlja vlogo *JobTracker* in vlogo *sužnja TaskTracker*.

V diplomski nalogi se sklicujemo na dve vrsti vozlišč, in sicer na **nadzorno** in **delovno vozlišče**. Nadzorno vozlišče ima na voljo več virov (delovni pomnilnik, število jeder, diskovni prostor ...), zato nanj namestimo zahtevnejše vloge in storitve. Pri izdelavi diplomske naloge smo se odločili za naslednjo konfiguracijo. Vlogi *TaskTracker* in *DataNode* smo namestili na vsa vozlišča gruče, medtem ko smo vlogi *JobTracker* in *NameNode* namestili na eno nadzorno vozlišče. V produkcijskem okolju sta vlogi *JobTracker* in *NameNode* običajno nameščeni na ločenih vozliščih, a glede na to, da smo imeli v gruči eno občutno zmogljivejše vozlišče, smo nanj namestili obe vlogi.

	NameNode	DataNode	JobTracker	TaskTracker
Delovno vozlišče		x		x
Nadzorno vozlišče	x	x	x	x

Tabela 2.1: Prikaz možne konfiguracije vlog znotraj HDFS in Hadoop MapReduce glede na tip vozlišča.

2.3.4 YARN

V sklopu izdaje Hadoop 0.23 je MapReduce doživel korenite spremembe, kar je vodilo v nastanek modula YARN, ki mu pravimo tudi MapReduce 2. Poglavitna sprememba je bila razdelitev vloge *JobTracker* v več samostojnih demonov. Tako sta nastali komponenti *ResourceManager*, ki je globalna, in *ApplicationMaster*, ki lahko deluje v obsegu klasičnega posla MapReduce ali

acikličnega usmerjenega grafa (DAG) poslov [7].

V diplomskem delu se osredotočamo predvsem na vzpostavitev in konfiguracijo sistema Hadoop z namenom spoznavanja in učenja programiranja na podlagi programskega modela MapReduce. Ker prva različica Hadoop MapReduce povsem zadostuje navedenim potrebam, se v diplomskem delu nismo podrobno posvetili modulu YARN. Sicer so navodila v diplomskem delu prilagojena rabi prve izdaje MapReduce, a namesto nje lahko uporabnik uporabi tudi YARN. MapReduce posle, ki so predstavljeni v diplomskem delu je moč poganjati z uporabo YARN, saj slednji zagotavlja kompatibilnost s prvo izdajo MapReduce [7].

2.3.5 Cloudera, CDH in Cloudera Manager

Cloudera Inc. je podjetje s sedežem v Ameriki, ki se ukvarja z razvojem programske opreme, temelječe na platformi Hadoop. Podjetje ima pod svojim okriljem tudi enega izmed razvijalcev, ki je sodeloval pri sami zasnovi platforme Hadoop, Douga Cuttinga, ki od leta 2009 zaseda mesto vodje arhitekta programske opreme v podjetju.

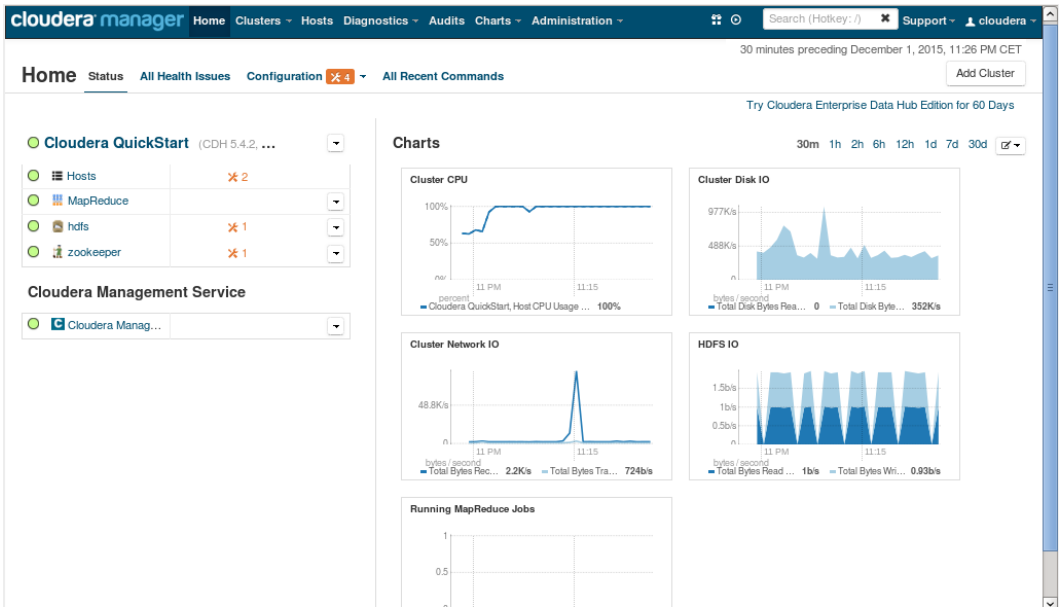
Podjetje nudi široko paleto produktov in storitev. Med produkte spadajo Cloudera Enterprise, Cloudera Navigator, Cloudera Director – če jih naštejemo samo nekaj. V okviru izdelave diplomske naloge smo uporabili programski rešitvi CDH in Cloudera Manager [14].

CDH oz. **Cloudera's distribution including Hadoop** spada med najbolj priljubljene distribucije sistema Apache Hadoop. Je odprtokodna in v celoti spada pod licenco podjetja Apache. CDH oskrbi uporabnika z vsemi osnovnimi elementi platforme Hadoop, porazdeljenim datotečnim sistemom in zmožnostjo vzporednega procesiranja podatkov skupaj z dodatnimi komponentami, kot so uporabniški vmesnik, zagotovljena varnost in možnost integracije s široko paleto strojne in programske opreme [16].

Pri izdelavi diplomske naloge je bila uporabljena različica CDH 5.4 (op. a. na voljo je že različica 5.5). Z različico CDH 5 je mogoča uporaba MapReduce 2 oz. YARN, a podjetje še vedno zagotavlja kompatibilnost s prvo različico

MapReduce [21]. Ravno kompatibilnost z obema različicama MapReduce je bil razlog za uporabo in predstavitev CDH 5. Čeprav navodila zajemajo uporabo prve različice MapReduce, je uporabnikom, ki se v bodoče želijo spoznati z YARN, ta možnost še vedno odprta.

Cloudera Manager je programska oprema, ki je v sožitju s CDH namenjena za upravljanje gruč naprav, ki vršijo posle MapReduce. Omogoča avtomatizirano nameščanje sistemskih paketov, skrajša čas namestitve, zagotavlja pregled stanja vozlišč in storitev v realnem času, nudi orodja za diagnostiko gruč in ostalo. Skratka, Cloudera Manager nudi centralizirano mesto nadzora in upravljanja z vozlišči in gručami, zaradi česar sta nadzor in upravljanje sistema preprosta in učinkovita [22]. Uporabili smo različico Cloudera Manager 5.



Slika 2.6: Zajem zaslona domače strani Cloudera Manager.

Poglavje 3

Vzpostavitev in uporaba sistema Hadoop

3.1 Quickstart Cloudera VM

Da bi čim bolj poenostavili namestitev in vzpostavitev porazdeljenega sistema Hadoop, smo za izhodišče uporabili vnaprej pripravljeno virtualno napravo QuickStart, ki jo lahko prenesemo z uradne spletne strani podjetja Cloudera na naslednji povezavi – `http://www.cloudera.com/content/cloudera/en/downloads/quickstart_vms/cdh-5-4-x.html`.
Na virtualni napravi teče operacijski sistem CentOS 6.4. Mogoče jo je poganjati v virtualnih okoljih VMware, KVM in VirtualBox. Virtualna naprava ima nameščen tako CDH kot Cloudera Manager in je pripravljena za takojšnjo uporabo [23].

3.2 Sistemske zahteve

Platformo Hadoop je s pomočjo Cloudere moč vzpostaviti na več različnih načinov z ozirom na to, kakšne vrste strojna oprema nam je na voljo. Uporabili smo opcijo Cloudera Express [24], ki jo sestavljata Cloudera Manager in CDH. Poganjanje virtualne naprave za svoje delovanje potrebuje gostitelja,

ki ima procesor s 64-bitno arhitekturo, najmanj 8 GB delovnega pomnilnika in enega od podprtih virtualnih okolij (VirtualBox, VMware, KVM) [23]. Vozlišče, na katero smo namestili Cloudera Express, imenujemo **nadzorno vozlišče**, medtem ko ostala vozlišča gruče imenujemo **delovna vozlišča**. Sistemske zahteve glede na tip vozlišča so naslednje:

1. Nadzorno vozlišče:
    - minimalne zahteve: 8 GB delovnega pomnilnika, dve virtualni jedri procesorja s 64-bitno arhitekturo in 25 GB diskovnega prostora,
    - priporočena konfiguracija: 10 GB delovnega pomnilnika, štiri virtualna jedra procesorja s 64-bitno arhitekturo in 64 GB ali več diskovnega prostora.
2. Delovno vozlišče:
    - minimalne zahteve: 2 GB delovnega pomnilnika, eno virtualno jedro procesorja s 64-bitno arhitekturo in 20 GB diskovnega prostora,
    - priporočena konfiguracija: 4 GB delovnega pomnilnika, dve virtualni jedri virtualnega procesorja s 64-bitno arhitekturo in 25 GB ali več diskovnega prostora.

Tako je možno na nekoliko zmogljivejšem računalniku, ki ima na voljo 16 GB delovnega pomnilnika in dvojedrno centralno procesno enoto s štirimi logičnimi nitmi, brez težav poganjati eno nadzorno vozlišče in dve delovni. Tovrstna konfiguracija že uspešno simulira porazdeljeni sistem.

V kolikor uporabnik nima na voljo računalnika, ki bi zmogel poganjati gručo dveh ali več vozlišč, vključno z nadzornim, lahko gručo pripravi na več osebnih računalnikih, ki bodo poganjali virtualne naprave. Ob tem mora le eden od njih ustrezati minimalnim sistemskim zahtevam nadzornega vozlišča, medtem ko lahko ostali računalniki ustrezajo zahtevam za vzpostavitev delovnih vozlišč. To pomeni, da lahko v primeru, ko imamo na voljo dva

računalnika, ki imata vsaj osem GB delovnega pomnilnika, vzpostavimo eno nadzorno vozlišče in tri delovna vozlišča.

3.3 Zahteve omrežnih nastavitev

Cloudera Manager za potrebe naslavljanja vozlišč v gruči potrebuje urejeno preslikavo med **naslovi IP** vozlišč in njihovimi imeni (**hostname**). V ta namen morajo imeti vozlišča pravilno nastavljeno datoteko **/etc/hosts**, ki mora biti konsistentna na vseh vozliščih gruče. Datoteka ne sme vsebovati podvojenih naslovov IP. Imena vozlišč ne smejo vsebovati velikih črk [18].

Navodila, ki smo jih v nadaljevanju zapisali za vzpostavitev nadzornega in delovnega vozlišča, predvidevajo, da so vsa vozlišča znotraj istega omrežja in povezana na isti omrežni usmerjevalnik. Tako je najlažje vzpostaviti povezavo med vozlišči in omrežnim vmesnikom virtualnih naprav nastaviti statične naslove IP, saj omrežni usmerjevalnik neposredno naslavlja vozlišča na podlagi njihovih naslovov MAC.

V navodilih smo opisali tudi scenarij, kako vzpostaviti gručo virtualnih naprav na enem gostitelju, ko uporabnik nima možnosti spreminjanja nastavitev na omrežnem usmerjevalniku oziroma nima dostopa do njega.

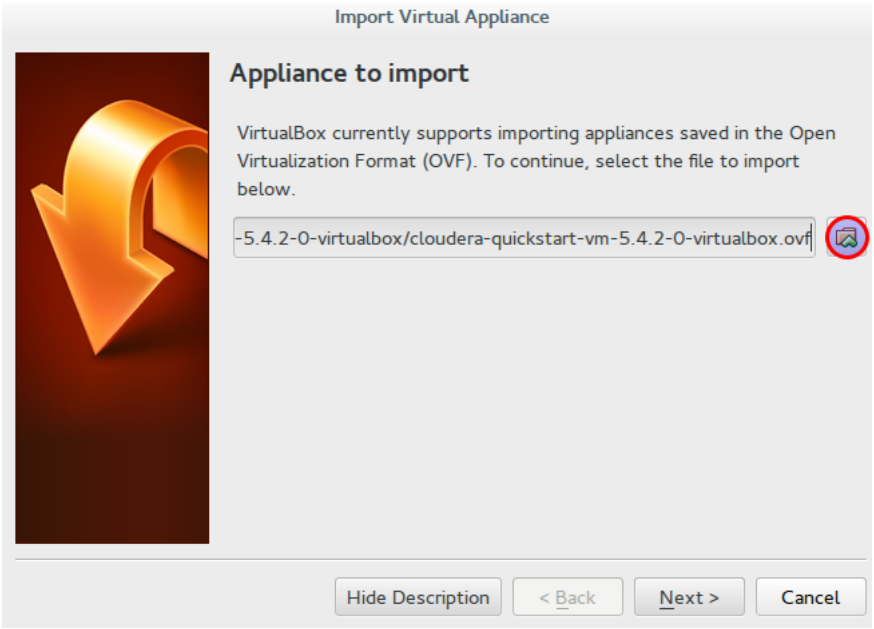
3.4 Vzpostavitev nadzornega vozlišča

V tem podpoglavju je predstavljen postopek, kako zagnati virtualno napravo QuickStart, ki lahko služi kot psevdogruča, saj jo sestavlja le eno vozlišče. Čeprav gre za psevdogručo, lahko z njeno pomočjo brez težav poganjamo zahtevke MapReduce. Hkrati nam je to vozlišče služilo tudi kot nadzorno vozlišče, ko smo v gručo dodali ostala delovna vozlišča. Navodila za njihovo vzpostavitev so predstavljena v naslednjem poglavju.

1. Z uradne Clouderine spletne strani prenesemo virtualno napravo Quickstart Cloudera. Možnost za prenos virtualne naprave se nahaja na: `http://www.cloudera.com/content/cloudera/en/downloads/`

`quickstart_vms/cdh-5-4-x.html`. Dotična virtualna naprava je prilagojena za poganjanje znotraj virtualnega okolja VirtualBox. Na spletni strani lahko uporabnik izbira med dvema virtualnima napravama, ki se razlikujeta v nameščeni verziji CDH. Na voljo sta CDH 5.4 in CDH 5.5.

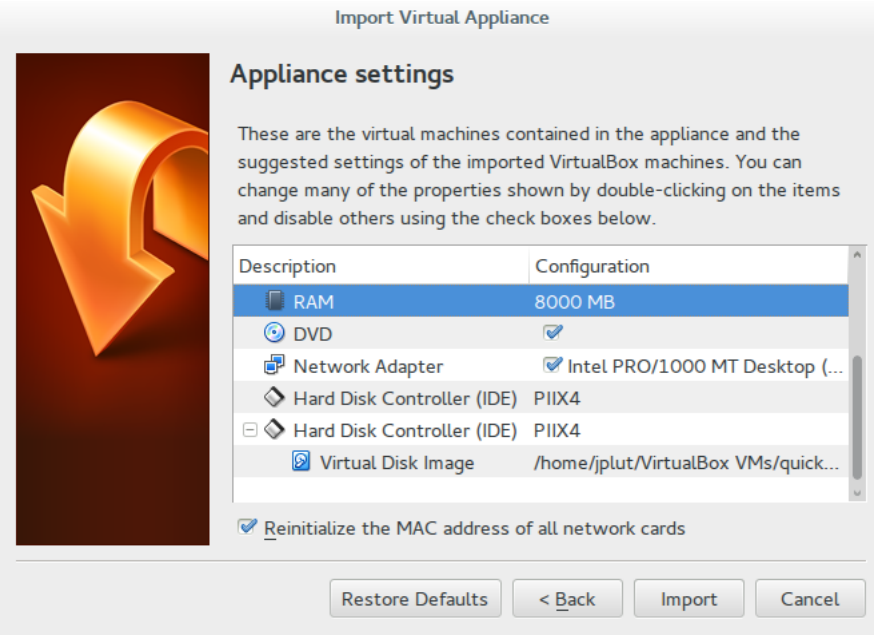
2. Ko se prenos konča, datoteko najprej razširimo. Nato virtualno napravo uvozimo v virtualizacijsko okolje VirtualBox. To storimo s klikom na **File** v vrstičnem meniju in izbiro možnosti **Import Appliance** v okolju VirtualBox. V pojavnem oknu s klikom na ikono v obliki mape z zeleno puščico odpremo dialog za odpiranje datotek in izberemo datoteko s končnico **ovf**, ki se nahaja v mapi, kjer smo preneseno datoteko razširili.



Slika 3.1: Dialog za uvoz virtualne naprave.

3. V oknu za nastavljanje lastnosti strojne opreme virtualne naprave nastavimo želene vrednosti. Nadzornemu vozlišču dodelimo vsaj 8 GB delovnega pomnilnika in dve jedri virtualnega procesorja. Nato izberemo opcijo "Reinitialize the MAC address of all network cards", ki

ponastavi naslov MAC virtualne naprave - s tem zagotovimo, da ima nova virtualna naprava drugačen naslov MAC od ostalih. Ko končamo z urejanjem nastavitev, kliknemo na gumb **Import**, ki sproži postopek uvoza virtualne naprave.

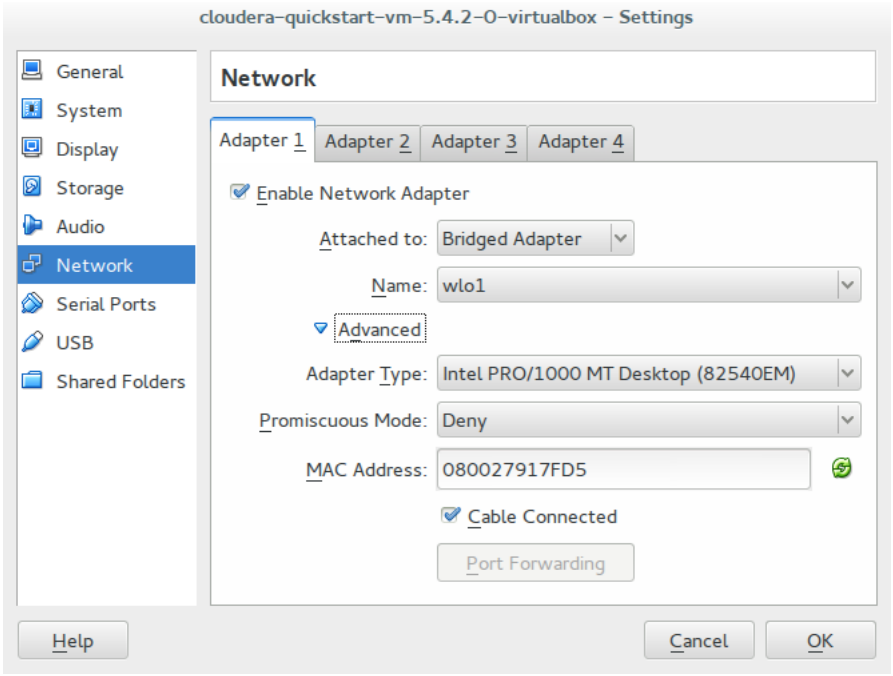


Slika 3.2: Dialog za nastavitev strojne opreme virtualne naprave.

4. Ko se uvoz virtualne naprave konča, ji določimo tip omrežnega vmesnika. Izberimo tip **Bridged Adapter**. S tem navideznemu omrežnemu vmesniku omogočimo neposreden dostop do omrežnega usmerjevalnika, saj se le-ta poveže neposredno na omrežni vmesnik gostitelja [15].

   Do nastavitev omrežnega vmesnika dostopamo tako, da v levem meniju okolja VirtualBox z desnim klikom izberemo pravkar uvoženo virtualno napravo, iz spustnega seznama izberemo možnost **Settings** in kliknemo na ikono z oznako **Network**. V zavihku z naslovom **Adapter 1** obkljukamo možnost **Enable Network Adapter**, ki aktivira omrežni vmesnik, in pri opciji **Attached to:** izberemo možnost **Bridged Adapter**. Nato kliknemo na gumb **Advanced**, ki prikaže doda-

tne možnosti za nastavitve, med drugim tudi naslov MAC, ki si ga na tem mestu zapišemo, saj ga bomo v naslednjih korakih potrebovali pri konfiguraciji. Spremembe shranimo s klikom na gumb **OK**.



Slika 3.3: Nastavitve omrežnega vmesnika

.

5. Na usmerjevalniku nastavimo statičen naslov IP, ki je vezan na naslov MAC omrežnega vmesnika virtualne naprave. S tem zagotovimo, da se naslov IP omrežnega vmesnika ne spreminja. Slednje je potrebno, če želimo urediti preslikovanje med imeni vozlišč (hostname) in njihovimi naslovi IP.

6. Virtualno napravo zaženemo tako, da v levem meniju okolja VirtualBox izberemo uvoženo napravo in kliknemo na ikono **Start** v zgornjem levem kotu.

7. Ko se virtualna naprava zažene, je najprej potrebno urediti nastavitvene datoteke omrežnih vmesnikov. Z ukazom `ifconfig -a` izpišemo

vse omrežne vmesnike sistema. Izpišeta se dva omrežna vmesnika, in sicer **lo** in **eth1**. Nato uredimo nastavitveno datoteko **/etc/sysconfig/network-scripts/ifcfg-eth1**, ki hrani nastavitve omrežnega vmesnika **eth1**.

Primer nastavitvene datoteke:

```
DEVICE=eth1
BOOTPROTO=dhcp
HWADDR=08:00:27:91:7F:D5
NM_CONTROLLED=yes
ON_BOOT=yes
TYPE=Ethernet
```

Prek teh nastavitev sistemu sporočimo, da omrežni vmesnik z imenom **eth1** dobi naslov IP prek omreženega protokola **DHCP**, da z njim upravlja storitev **Network Manager**, da je omrežni vmesnik ob zagonu aktiven in da naprava uporablja **Ethernet**. V parameter **HWADDR** vnesemo naslov MAC omrežnega vmesnika. V kolikor si ga nismo zapisali ali zapomnili tekom nameščanja virtualne naprave, ga lahko preverimo z ukazom `ifconfig -a`.

Ko datoteko shranimo, poženemo ukaz
`sudo service network restart`,
ki ponovno zažene omrežne vmesnike s pravkar vnesenimi nastavitvami. Ko se omrežni vmesniki vzpostavijo, preverimo, ali ima virtualna naprava delujočo povezavo s spletom. To storimo z ukazom `ping 8.8.8.8`, ki pošilja pakete na Googlove strežnike DNS. V kolikor se povezava ne vzpostavi, preverimo, ali smo med vnašanjem nastavitev vmesnikov vnesli napačne vrednosti.

8. Na tem mestu smo pripravili vse potrebno za zagon storitve Cloudera Manager. Storitev zaženemo prek ukazne vrstice z naslednjim ukazom: `/home/cloudera/cloudera-manager --express --force`

9. Ko se vse storitve zaženejo in inicializirajo, odpremo brskalnik in se prijavimo v Cloudera Manager. Storitev teče na naslovu `quickstart.cloudera:7180`. Za vstop uporabimo uporabniško ime **cloudera** in njemu enako geslo.

10. Virtualna naprava QuickStart ima privzeto nameščeno paleto storitev podjetja Apache, ki uporabniku nudi celostno okolje za razvoj. Glede na to, da za naše potrebe ne potrebujemo vseh nameščenih storitev, ohranimo le storitvi **hdfs** (porazdeljeni datotečni sistem in **zookeeper** – centralizirana storitev, ki skrbi za informacije o konfiguraciji vozlišča, njegovi sinhronizaciji, imenskem prostoru idr. [26]). S tem tudi sprostimo vire virtualne naprave.

    Navodila ne predvidevajo rabe večine nameščenih storitev, zato jih lahko ustavimo. To storimo tako, da v spustnem seznamu poleg imena storitve izberemo možnost **Stop**. Če želimo poljubno storitev odstraniti, kliknemo na spustni seznam poleg imena storitve in izberemo opcijo **delete**. Ker so storitve medsebojno odvisne, jih je potrebno odstraniti v naslednjem vrstnem redu: **hue** (grafični vmesnik za analizo podatkov znotraj Hadoop) **impala**, **sqoop** (orodje za prenašanje podatkov med podatkovnimi bazami in Hadoopovim porazdeljenim sistemom HDFS), **oozie** (javanska aplikacija za časovno razporejanje poslov MapReduce), **hive** (podatkovno skladišče, ki v sodelovanju s Hadoopom nudi iskanje in analizo po velikih množicah podatkov, shranjenih v HDFS), **spark**, **Key-Value Store indexer** (storitev za indeksiranje vnosov v tabele znotraj HBase), **solr** (mehanizem za iskanje), **HBase** (skalabilna, porazdeljena podatkovna baza, ki podpira shranjevanje strukturiranih podatkov v velike tabele) in **YARN** (MapReduce 2), saj smo namesto njega uporabili kar **MapReduce**. Slednjega dodamo s klikom na spustni seznam in izbiro **Add a Service**. S seznama izberemo **MapReduce**, ki ga namestimo s privzetimi nastavitvami.

Tako smo vzpostavili gručo, sestavljeno iz le enega nadzornega vozlišča, ki

je pripravljena za osnovno uporabo. Glede na tovrstno konfiguracijo gruče obstaja verjetnost, da uporabnik prejema obvestila in opozorila s strani storitve Cloudera Manager, da katera od storitev na gruči ne deluje dobro oz. da prihaja do napak. Na primer, velika verjetnost je, da bo storitev HDFS javljala napako, da ne more ustvariti zadostnega števila repliciranih blokov, saj za svoje delovanje potrebuje vsaj tri vozlišča z nameščeno vlogo **DataNode** in podobno. Cloudera Manager javi napako in njen vzrok (sporoči tudi, kako napako odpraviti), tako da je nekatere napake moč odpraviti le z nekaj kliki. Navkljub nekaterim napakam, ki se lahko pojavijo, je na sistemu mogoče poganjati posle MapReduce. Tako vozlišče predstavlja začetno točko za vse, ki se želijo spoznati s sistemom Hadoop, saj omogoča njegovo uporabo v celoti.

3.5 Vzpostavitev in dodajanje delovnega vozlišča v gručo

Ko smo namestili in vzpostavili nadzorno vozlišče, smo se lotili dodajanja delovnega vozlišča, ki sta skupaj z nadzornim tvorila gručo. Obstaja več možnosti vzpostavitve delovnega vozlišča. Glede na to, da je bil naš cilj pri vzpostavitvi porazdeljenega sistema postaviti delujoč sistem čim hitreje in kar se da enostavno, smo se odločili, da bomo za delovno vozlišče uporabili vnaprej pripravljeno virtualno napravo, ki že ima nameščen operacijski sistem CentOS 6.4.

3.5.1 Vzpostavitev delovnega vozlišča

1. S strani `http://sourceforge.net/projects/virtualappliances/files/Linux/CentOS/CentOS-6.4-amd64-minimal.ova/download` prenesemo sliko virtualne naprave (ang. virtual applience), ki ima prednameščen operacijski sistem CentOS 6.4. Tako skrajšamo čas vzpostavitve sistema, saj nam ni potrebno nameščati operacijskega sistema in

sistemskih paketov na novo virtualno napravo. Virtualna naprava je prilagojena za poganjanje znotraj virtualnega okolja VirtualBox.

V kolikor bi uporabljali virtualno okolje VMWare, bi prednameščeno virtualno napravo prenesli s strani `http://virtual-machine.org/centos-6-x86_64-64bit-vmware-image-download`. Če bi želeli poganjati virtualno napravo v virtualnem okolju KVM, pa bi bilo po vsej verjetnosti potrebno virtualno napravo vzpostaviti ročno, saj na spletu niso prosto dostopne oziroma nam jih ni uspelo najti.

2. Virtualno napravo uvozimo v virtualno okolje VirtualBox, ji nastavimo količino delovnega pomnilnika, število procesorskih enot, ponastavimo naslov MAC in nastavimo omrežni vmesnik ter ji na omrežnem usmerjevalniku nastavimo statični naslov IP. Vse naštete korake smo že opisali v poglavju 3.4, in sicer od drugega do osmega koraka. Virtualna naprava ima registrirana dva uporabnika, in sicer uporabnika **root** z geslom **toor** in uporabnika **user** z geslom **nimda**. Za potrebe zagotavljanja varnosti uporabnikoma zamenjamo gesli.

3. Virtualni napravi nastavimo ime **hostname**. Ime naprave določimo prek terminalske vrstice. Ker želimo, da spremenjeno ime naprave ostane shranjeno tudi ob naslednjem zagonu naprave, uredimo konfiguracijsko datoteko **/etc/sysconfig/network**, ki bo poskrbela, da naše nastavitve ostanejo shranjene.

   Primer ukaza za spremembo imena naprave:

   ```
   hostname hadoop2.example.com
   ```

   Primer nastavitvene datoteke **/etc/sysconfig/network**:

   ```
   NETWORKING=YES
   HOSTNAME=hadoop2.example.com
   ```

4. Da omogočimo komunikacijo med vozlišči, je potrebno pravilno nastaviti konfiguracijsko datoteko **/etc/hosts**. Vanjo vpišemo vsa vozlišča,

ki jih povezujemo v gručo. Datoteko **/etc/hosts** je potrebno nastaviti na **vseh vozliščih**, tako na nadzornem kot na delovnih, saj s tem omogočimo komunikacijo med njimi prek uporabe imen *hostname*. Namreč, vsako od vozlišč mora znati pretvoriti ime *hostname* v naslov IP vozlišča in obratno [18].

```
127.0.0.1      localhost  localhost.localdomain
192.168.1.121  quickstart.cloudera
192.168.1.123  hadoop2.example.com
```

5. V uradni dokumentaciji na spletni strani podjetja Cloudera so navedeni koraki, ki jih je potrebno izpolniti pred vzpostavitvijo gruče. Na vozliščih, ki jih povezujemo v gruče, je potrebno ustaviti požarni zid **iptables** in modul za zaščito **SELinux**. Razlog za to je, da oba pogosto povzročata motnje v komunikaciji med vozlišči [25].

   Najprej shranimo obstoječe nastavitve požarnega zidu in ga nato ugasnemo.

   ```
   iptables-save > /root/firewall.rules
   chkconfig iptables off
   /etc/init.d/iptables stop
   ```

   Nato še onemogočimo varnostni modul SELinux. V datoteki **/etc/sysconfig/selinux/** parameter **SELINUX** nastavimo na **permissive**. Z ukazom `setenforce 0` sporočimo sistemu, da želimo uveljaviti spremembo konfiguracije varnostnega modula [25].

6. Nastaviti je potrebno tudi parameter **swappiness**. Z njim napravi sporočimo, kako pogosto naj prenaša strani iz delovnega pomnilnika na trdi disk. Parametru je moč nastaviti vrednost na intervalu od 0 in 100. Privzeta vrednost parametra je 60, medtem ko je za porazdeljeni sistem Hadoop priporočljivo, da je parameter nastavljen na manjšo vrednost, saj s tem izboljšamo stabilnost in učinkovitost izvajanja sistema [27].

   Parameter *swappiness* spremenimo z ukazom:

```
sysctl vm.swappiness=10
```

Ker želimo, da sprememba parametra ostane shranjena tudi ob ponovnih zagonih virtualne naprave, v konfiguracijski datoteki **/etc/sysctl.conf** nastavimo vrednost parametra **swappiness**.

```
vm.swappiness=10
```

7. Ko končamo z dodajanjem vozlišča v gručo, se na nadzornem vozlišču prek povezave **quickstart.cloudera:7180** prijavimo v Cloudera Manager in dodelimo vloge novemu delovnemu vozlišču. Na novo dodanemu vozlišču dodamo vlogo **DataNode** znotraj **HDFS** storitve. Znotraj storitve **MapReduce** vozlišču dodelimo vlogo **TaskManager**.

3.5.2 Dodajanje delovnih vozlišč v gručo

Ko smo nastavili nadzorno in vsaj eno delovno vozlišče, smo delovno vozlišče vključili v gručo.

1. Pri dodajanju delovnega vozlišča v gručo se prijavimo v storitev Cloudera Manager. Do storitve dostopamo prek brskalnika na nadzornem vozlišču. V naslovno vrstico vpišemo naslov url **quickstart.cloudera:7180**, ki odpre domačo stran storitve Cloudera Manager. Vanjo se vpišemo z uporabniškim imenom **cloudera** in njemu enakim geslom.

2. Nato pričnemo s postopkom dodajanja novih vozlišč v gručo. Na domači strani storitve Cloudera Manager v vrhnjem meniju kliknemo na izbiro **Hosts**. Na naslednji strani kliknemo na gumb **Add New Hosts to Cluser**. Pri načinu dodajanja novih vozlišč v gručo izberemo možnost **Classic Wizard**, saj je bilo za drugo opcijo potrebno imeti nameščeno storitev **Cloudera Director**, ki pa je nismo namestili. Na naslednji strani nas sistem obvesti o različnih možnostih dodajanja novih vozlišč v gručo. S klikom na gumb **Continue** smo preusmerjeni na stran z vnosnim poljem, kjer vpišemo imena vozlišč, tj. **hostname**,

ki jih dodajamo v gručo. Virtualne naprave je mogoče naslavljati tudi prek naslova IP. Cloudera Manager omogoča dodajanje vozlišč s pomočjo vzorcev, na primer, če želimo v gručo dodati vozlišča **quickstart1.cloudera**, **quickstart2.cloudera** in **quickstart2.cloudera**, preprosto vpišemo **quickstart[1-3].cloudera** in kliknemo gumb **Search**. Sistem bo samodejno prepoznal dosegljiva vozlišča. Ostale namige in priporočila glede naslavljanja s pomočjo vzorcev je moč prebrati na povezavi **Patterns**. S klikom na gumb **Search** Cloudera Manager samodejno izbere vsa dosegljiva vozlišča, ki ustrezajo podanemu vzorcu in še niso povezana v gručo. Postopek dodajanja nadaljujemo s klikom na gumb **Continue**.

3. Na naslednji strani izberemo izdajo CDH, ki jo nameščamo na vozlišče. Izberemo prvo izbiro – "Latest Release of CDH 5 compatible with this version of Cloudera Manager", saj tako nameščamo tiste pakete, ki so kompatibilni z izdajo Cloudera Manager, nameščeno na nadzornem vozlišču. Prav tako na dnu strani označimo izbiro **Matched release for this Cloudera Manager Server**, ki se nanaša na različico storitve **Cloudera Manager Agent**, ki je nameščena na delovno vozlišče in je kompatibilna z različico **Cloudera Manager Server**.

4. Na naslednji strani so izpisani pogoji uporabe. Na dnu strani obkljukamo možnost "Install Oracle Java Se Development Kit (JDK)", ki bo na vozlišče namestila orodja za razvijalce v programskem jeziku Java.

5. Na naslednji strani vpišemo uporabniško ime uporabnika, ki obstaja na delovnem vozlišču, prek katerega se Cloudera Manager znotraj seje SSH poveže na delovno vozlišče in namesti potrebne sistemske pakete. Izpolnimo polja za uporabniško ime, geslo in potrditev le-tega. Uporabniško ime in geslo morata pripadati uporabniku, ki je že registriran na vozlišču, ki ga dodajamo. V gručo je možno dodati več vozlišč hkrati, v kolikor imajo ta enakega uporabnika in uporabniško geslo. S klikom na gumb "Continue" se prične nameščanje paketov na izbrano

vozlišče. Ko je nameščanje končano, kliknemo "Continue".

6. Na tem mestu bo sistem preveril, ali se vozlišče, ki ga dodajamo, v konfiguraciji razlikuje s tistimi, ki so že v gruči. Cloudera Manager preveri, ali ima novo dodano vozlišče nastavljeno enako časovno cono, uro in datum kot ostala vozlišča, preveri datoteko **/etc/hosts**, opozori uporabnika, če ima novo dodano vozlišče previsoko nastavitev vrednosti parametra **swapiness** in predlaga najbolj optimalno ter mnogo drugega. V primeru, da se pojavljajo napake ali opozorila, sistem to navede in hkrati predlaga rešitev. Preverjanje lastnosti dodanega vozlišča je možno pognati poljubnokrat, vse dokler ne odpravimo nekonsistenc.

7. V tem koraku določimo vloge, ki jih bo novo delovno vozlišče poganjalo. V kolikor imamo vnaprej pripravljeno predlogo dodeljevanja storitev, jo lahko uporabimo. V nasprotnem primeru kliknemo "Continue" in Cloudera bo namestila storitve, ki so že nameščene na gruči – v našem primeru sta to HDFS in MapReduce.

8. Ko je vozlišče dodano gruči, na domači strani sistema Cloudera Manager izberemo storitev, katere vlogo želimo pripisati vozlišču, nato izberemo možnost "instances" ter kliknemo gumb "Add Role Instances". Prikaže se stran, kjer lahko novo dodanemu vozlišču dodelimo vlogo, npr. če se nahajamo znotraj storitve HDFS, lahko novo dodanemu vozlišču dodamo vlogo *DataNode*, glede na to, da naj bi imelo vsako vozlišče znotraj gruče nameščeno vlogo *DataNode*.

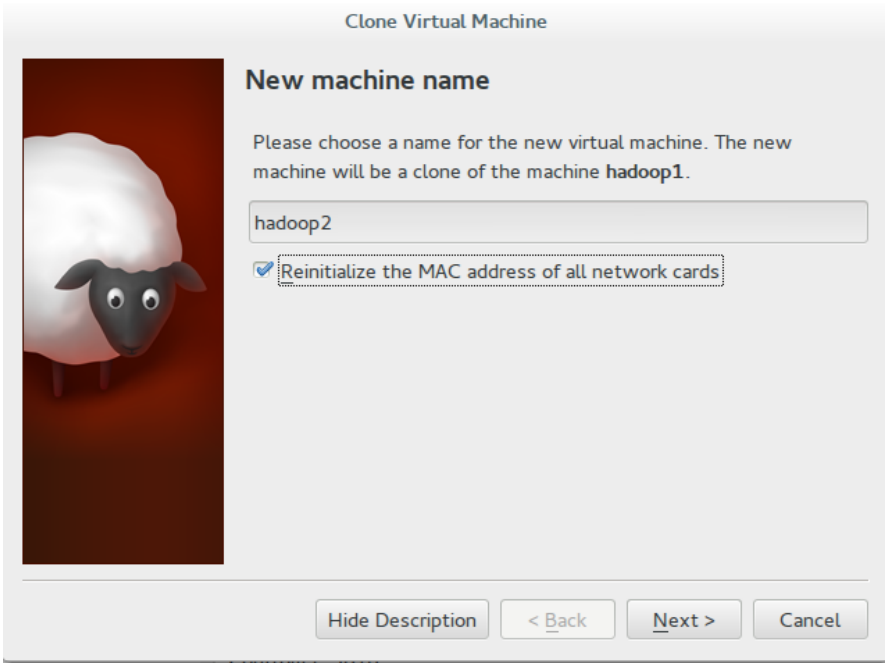
3.6 Razširjanje gruče

Ko smo vzpostavili tako nadzorno kot delovno vozlišče, smo se odločili razširiti našo gručo. Za izhodišče smo uporabili virtualno napravo delovnega vozlišča, ki smo ga s pomočjo okolja VirtualBox klonirali in prilagodili potrebne parametre, ki so specifični za vsako od virtualnih naprav posebej.

1. Vozlišče, ki ga želimo klonirati, ugasnemo, saj virtualne naprave v delovanju ni mogoče klonirati. Virtualno napravo ugasnemo prek ukazne vrstice z ukazom `poweroff`.

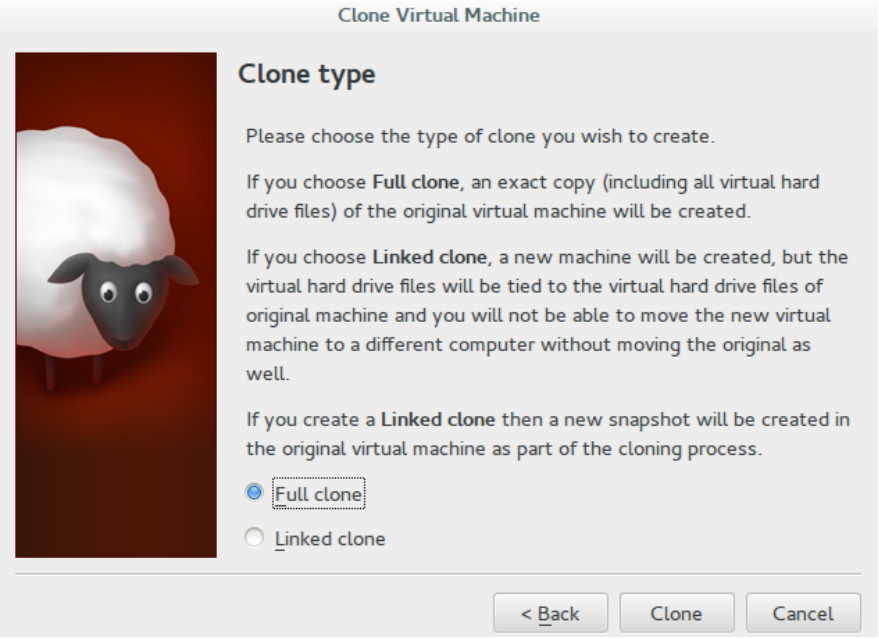
2. V levem meniju okolja VirtualBox izberemo virtualno napravo, ki jo želimo klonirati. Nanjo kliknemo z desnim miškinim gumbom in iz spustnega seznama izberemo opcijo **Clone**.

3. Pojavi se okno za nastavitev imena virtualne naprave. V vnosno polje vnesemo želeno ime. Pod vnosnim poljem izberemo opcijo **Reinitialize the MAC address of all network cards**, ki bo klonirani virtualni napravi določila nove naslove MAC omrežnih vmesnikov. Slednje je potrebno, ker nočemo, da bi imela nova virtualna naprava enake strojne naslove kot njen original. Posledično ne bi bilo mogoče nastaviti statičnega naslova IP nove naprave, saj bi imela enak strojni naslov kot original.



Slika 3.4: Kloniranje virtualne naprave.

4. Ker želimo ustvariti vozlišče, ki ga je možno prenesti na več fizičnih računalnikov in je neodvisno od svojega originala, v naslednjem oknu izberemo opcijo **Full clone**. Nato kliknemo na gumb **Clone**, ki sproži proces kloniranja.



Slika 3.5: Izbira načina kloniranja.

5. Ko je virtualna naprava klonirana, jo izberemo v glavnem meniju okolja VirtualBox in jo zaženemo s klikom na ikono **Start**.

6. Na usmerjevalniku novo ustvarjeni virtualni napravi nastavimo statični naslov IP glede na njen strojni (MAC) naslov. V kolikor strojnega naslova ne vemo, z ukazom `ifconfig -a` izpišemo vse omrežne vmesnike in njihove lastnosti. Med njimi je izpisan tudi naslov MAC.

7. Potrebno je prilagoditi tudi nastavitveno datoteko omrežnega vmesnika. Ker ima novo ustvarjeno vozlišče drugačen strojni naslov, ga je potrebno spremeniti v datoteki **/etc/sysconfig/network-scripts /ifcfg-eth1**.

Primer nastavitvene datoteke:

```
DEVICE=eth1
BOOTPROTO=dhcp
HWADDR=08:00:27:91:2F:44
NM_CONTROLLED=yes
ON_BOOT=yes
TYPE=Ethernet
```

Prek teh nastavitev sistemu sporočimo, da omrežni vmesnik z imenom **eth1** dobi naslov IP prek omreženega protokola **DHCP**. Določa, da z njim upravlja storitev **Network Manager** in da je omrežni vmesnik ob zagonu naprave aktiven ter da naprava uporablja **Ethernet**. V parameter **HWADDR** je vnesen naslov MAC omrežnega vmesnika.

Nato ponovno zaženemo vse omrežne vmesnike z ukazom:

`service network restart`.

S pošiljanjem paketkov na Googlove strežnike DNS preverimo, ali imamo delujočo povezavo

`ping 8.8.8.8`.

8. Ker gre za klon že obstoječe virtualne naprave, je potrebno namestiti določene parametre in spremenljivke, ki so lastne vsakemu vozlišču posebej. Zamenjati je potrebno ime naprave (hostname), saj trenutno v gruči že imamo napravo z enakim imenom. Primer ukaza za spremembo imena naprave:

   ```
   hostname hadoop3.example.com
   ```

   Potrebno je spremeniti tudi datoteko **/etc/sysconfig/network**, ki ob zagonu virtualne naprave nastavi ime. Primer nastavitvene datoteke:

   ```
   NETWORKING=YES
   HOSTNAME=hadoop2.example.com
   ```

9. Novo ustvarjenemu vozlišču je potrebno urediti datoteko **/etc/hosts**, ki mora vsebovati naslov IP in ime naprave tako novo dodanega vozlišča kot tudi **vseh ostalih** vozlišč v gruči. Na tem mestu popravimo vsebino nastavitvene datoteke tudi na ostalih vozliščih v gruči, saj Cloudera Manager potrebuje konsistentno urejene preslikave med imeni naprav in naslovi IP na vsakem od vozlišč gruče.

   Primer pravilno urejene datoteke **/etc/hosts**:

   ```
   127.0.0.1       localhost localhost.localdomain
   192.168.1.101   hadoop1.example.com hadoop1
   192.168.1.102   hadoop2.example.com hadoop2
   192.168.1.103   hadoop3.example.com hadoop3
   192.168.1.120   quickstart.cloudera quickstart
   ```

10. Ker smo klonirali virtualno napravo, ki že ima nameščeno vse potrebno za poganjanje poslov MapReduce v sklopu platforme Hadoop, je potrebno le še spremeniti ID vozlišča (**host id**), ki ga uporablja Cloudera Manager, saj je enak vozlišču, katerega kopijo smo ustvarili. Parameter naj predstavlja poljuben niz. Nastavimo ga tako, da v nastavitveni datoteki **/etc/default/cloudera-scm-agent** nastavimo vrednost parametru `CMF_AGENT_ARGS`.

    Primer vrstice nastavitvene datoteke z nastavljenim parametrom:

    ```
    CMF_AGENT_ARGS="--host_id=00000"
    ```

    Ko parameter nastavimo, ponovno zaženemo Clouderinega agenta, ki bo uveljavil nove nastavitve.

    ```
    service cloudera-scm-agent restart
    ```

11. Na tem mestu je vozlišče pripravljeno za dodajanje v gručo. Vozlišče dodamo v gručo po korakih, zapisanih v 3.5.2.

Kloniranje virtualne naprave je priporočljivo za tista vozlišča, ki še nimajo dodeljenih DataNode, TaskTracker in ostalih vlog, saj se pri kloniranju prepišejo njihovi identifikatorji, kar povzroča težave v delovanju gruče. V izogib slednjemu kloniramo vozlišča, ki nimajo določenih vlog, oziroma ustavimo vloge in jih izbrišemo ter vozlišče odstranimo iz gruče in se šele nato lotimo kloniranja.

3.7 Napake in težave pri dodajanju novega vozlišča gruči in kako jih odpraviti

Če med dodajanjem novega vozlišča v gruče pride do kakršne koli napake, ki vodi do prekinitve nameščanja potrebnih sistemskih paketov, preverimo dnevnik (ang. logs). To storimo s klikom na povezavo **Details**. V njem je zapisano, v katerem koraku je bilo nameščanje prekinjeno in kaj je bil vzrok temu.

V kolikor nameščanje paketov obstane in izpiše status **"Acquiring installation lock"**, se prek seje SSH povežemo na dotično vozlišče in izbrišemo datoteko v `/tmp/.scm_prepare_node.lock`, ki preprečuje nadaljnje izvajanje inštalacije, prekinemo namestitev in jo ponovno zaženemo s klikom na gumb "Retry failed hosts". Do situacije z omenjenim sporočilom pride v primeru, ko je bilo predhodno nameščanje paktov prekinjeno s strani uporabnika. Po odstranitvi zaklepa lahko nadaljujemo z nameščanjem potrebnih sistemskih paketov.

V primeru, da se povezava med vozliščem in repozitorijem s sistemskimi paketi prekine, kliknemo na gumb **Details** in preverimo, v katerem od korakov je povezava padla in kateri od paketov ni bil nameščen do konca. Nato vzpostavimo sejo SSH z vozliščem, na katerem je padla povezava, in ročno namestimo paket, ki ni bil naložen. Točen ukaz za namestitev paketa najdemo v dnevniku zapisov oziroma s klikom na gumb **Details**. Ko ročno namestimo paket, ponovno poženemo namestitev na vozlišču, kjer je prišlo do težav. Včasih prihaja do težav s povezavo do repozitorija, tako da ob-

staja verjetnost, da je ta korak potrebno ponoviti nekajkrat tekom procesa dodajanja vozlišča.

Če se razlog za prekinitev dodajanja novega vozlišča glasi **No route to host**, pomeni, da nadzorno vozlišče ne more vzpostaviti seje SSH z delovnim vozliščem. Na tem mestu je potrebno preveriti, kakšne nastavitve imata nadzorno in delovno vozlišče. V kolikor gručo sestavljajo vozlišča, ki tečejo na ločenih gostiteljih, je potrebno preveriti, ali so vsi povezani na isti usmerjevalnik, ki jim statično določi naslov IP glede na njihov naslov MAC. Vozliščem nastavimo le en omrežni vmesnik, ki naj bo tipa **Bridged network**. Tako se bomo izognili velikemu številu morebitnih težav pri konfiguraciji omrežja.

3.8 Konfiguracija omrežnega vmesnika brez dostopa do omrežnega usmerjevalnika

Za uporabnike, ki nimajo dostopa do omrežnega usmerjevalnika, a imajo na voljo računalnik, ki je dovolj zmogljiv, da lahko poganja gručo, sestavljeno iz dveh ali več virtualnih naprav, smo pripravili napotke, kako nastaviti nastavitve omrežnih vmesnikov tako, da omogočajo komunikacijo med vozlišči. Prav tako smo opisali, kako nastaviti statičen naslov IP omrežnim vmesnikom brez posega v nastavitve omrežnega usmerjevalnika. Slednje smo realizirali z uporabo dveh virtualnih omrežnih vmesnikov na vsaki od virtualnih naprav. Prvi virtualni omrežni vmesnik tipa **NAT** poskrbi za dostop virtualne naprave do spleta. Drugi virtualni omrežni vmesnik tipa **Host-only adapter** pa omogoča komunikacijo med virtualnimi napravami, nameščenimi na istem gostitelju, in možnost nastavljanja statičnega naslova IP. Oboje je potrebno za vzpostavitev gruče na enem gostitelju.

1. V kolikor želimo uporabljati vrsto omrežnega vmesnika **Host-only adapter**, je potrebno pripraviti njegovo predlogo zanj. V okolju VirtualBox v vrstičnem meniju izberemo **File** in nato **Preferences**. V pojavnem meniju izberemo opcijo **Network**, kliknemo na zavihek **Host-**

only Networks in nato na ikono z zelenim simbolom plus. V oknu se pojavi nov vnos, ki ga uredimo tako, da kliknemo na ikono s sliko izvijača. V zavihku **Adapter** nastavimo obseg naslovov IP virtualnih naprav. V našem primeru polju **IPv4 Address** nastavimo vrednost **10.210.55.1**. Nato nastavimo masko omrežja v polju **IPv4 Network Mask** na **255.255.255.0** in pri tem pustimo ostali polji nedotaknjeni. V zavihku **DHCP Server** poskrbimo, da opcija **Enable Server ni** izbrana.

2. Običajno imajo virtualne naprave omogočen samo en omrežni vmesnik, zato je ostale potrebno ročno omogočiti in nastaviti.

   V levem meniju okolja VirtualBox izberemo virtualno napravo in kliknemo na ikono **Settings**. V levem meniju pojavnega okna izberemo **Network** in nato zavihek **Adapter 1** ter izberemo možnost **Enable Network Adapter**. S spustnega seznama **Attached to** izberemo možnost **NAT**. Enako storimo v zavihku **Adapter 2**, le da v tem primeru s seznama izberemo opcijo **Host-only Adapter** in s spustnega seznama **Name** izberemo konfiguracijo omrežnega vmesnika, ki smo ga nastavili v prejšnji točki – njegovo ime se začne z **vboxnet**, ki mu sledi zaporedna številka.

3. Ko končamo z nastavljanjem virtualnih omrežnih vmesnikov, je potrebno spremeniti še konfiguracijske datoteke znotraj virtualne naprave. Ker ima operacijski sistem CentOS 6 privzeto onemogočen omrežni vmesnik, slednjega omogočimo.

   V datoteki **etc/sysconfig/network-scripts/ifcfg-eth0** nastavimo, da ob zagonu naprave omogoči tudi omrežni vmesnik **eth0**. To storimo tako, da spremenljivko **ON_BOOT** nastavimo na **YES**.

   Primer spremenjene vrstice:

   ```
   ON_BOOT=yes
   ```

   Spremembe v konfiguracijski datoteki shranimo. Spremembe nastavitvene datoteke omrežnega vmesnika uveljavimo tako, da ponovno

zaženemo omrežni vmesnik. Slednje storimo z ukazom:
`sudo service network restart`

Ko se omrežni vmesnik ponovno zažene in inicializira, preverimo, ali imamo odprto povezavo s spletom. Z ukazom `ping 8.8.8.8`, ki pošilja paketke na Googlove srežnike DNS, preverimo, ali se paketki nemoteno pošiljajo.

4. Ko vzpostavimo delujočo povezavo s spletom, se lotimo konfiguracije drugega omrežnega vmesnika tipa **Host-only adapter**, ki omogoča nastavitev statičnega naslova IP virtualnih naprav, pa tudi komunikacijo med virtualnimi napravami na istem gostitelju. V tem primeru urejamo konfiguracijsko datoteko omrežnega vmesnika **eth1**.

   Opomba – številčenje omrežnih vmesnikov in njim ustreznih konfiguracijskih datotek se lahko razlikuje glede na napravo in njene namestitve. V navodilih predvidevamo, da sistem prepozna dva omrežna vmesnika in sicer **eth0** in **eth1**.

   Primer urejene konfiguracijske datoteke:

```
HWADDR=08:00:27:BB:D2:FA
DEVICE=eth1
TYPE=Ethernet
ON_BOOT=yes
NM_CONTROLLED=no
BOOTPROTO=static
IPADDR=10.210.55.101
NETMASK=255.255.255.0
```

   V konfiguracijsko datoteko vnesemo strojni naslov oz. naslov MAC (**HWADDR**) omrežnega vmesnika in njegovo ime (**DEVICE**). Nastavimo tip omrežnega vmesnika (**ETHERNET**) in prek spremenljivke (**NM CONTROLLED**) demonu **Network Manager** onemogočimo, da bi spreminjal nastavitve vmesnika. Nato še določimo naslov IP vmesnika in masko omrežja (**IPADDR** in **NETMASK**) ter nastavimo

način pridobivanja naslova IP (**BOOTPROTO**) - v našem primeru gre za statično naslavljanje. Pri urejanju konfiguracijske datoteke moramo biti pozorni, da virtualni napravi odredimo ustrezen naslov IP, saj mora biti v skladu z nastavitvami vmesnika **Host-only adapter**, ki smo ga predhodno nastavili. Naslov IP omrežnega vmesnika virtualne naprave mora biti v istem podmrežju, kot smo nastavili v nastavitvah **Host-only adapter** vmesnika v prejšnjih korakih.

Ko konfiguracijsko datoteko shranimo, ponovno zaženemo omrežne vmesnike. Z ukazom `ifconfig -a` preverimo, ali so se nastavitve omrežnih vmesnikov pravilno shranile. Na koncu preverimo, ali lahko z gostitelja, ki poganja virtualne naprave, z ukazom **ping** dosežemo pravkar konfigurirane omrežne vmesnike virtualne naprave.

3.9 Preizkus sistema

Ko smo vzpostavili gručo iz enega ali več vozlišč, je čas, da na vozlišča namestimo vloge. Vloge namestimo prek storitve Cloudera Manager.

1. Na nadzornem vozlišču se prijavimo v storitev Cloudera Manager. To storimo tako, da v naslovno vrstico brskalnika vnesemo naslov `quickstart.cloudera:7180`. V storitev se vpišemo z uporabniškim imenom **cloudera**. Geslo je enako uporabniškemu imenu. Zaradi zagotavljanja varnosti geslo ob prvi priložnosti spremenimo.

2. Na domači strani storitve Cloudera Manager so v levem meniju prikazane gruča in storitve, ki so v teku.

3. Vozliščem, ki so v gruči, dodamo vloge tako, da kliknemo na storitev, znotraj katere želimo dodati vlogo. Preusmerjeni smo na stran storitve, kjer so izpisane vse podrobnosti v zvezi z določeno storitvijo.

4. Vloge znotraj določene storitve, npr. *hdfs*, lahko dodajamo in odvzemamo tako, da v vrstičnem meniju kliknemo na izbiro **Instances**. Sis-

tem bo izpisal vse vloge, ki so v teku znotraj določene storitve. Tako bo na primer za storitev *hdfs* izpisal, na katerih vozliščih tečejo vloge *DataNode*, *NameNode* in ostale. S klikom na gumb **Add Role Instances** lahko prek preprostega čarovnika za dodajanje vlog enostavno namestimo želene vloge vozliščem znotraj gruče. Postopek za dodajanje vlog vozliščem je enak, ne glede na storitev, katere vlogo želimo dodati vozlišču.

5. Ko končamo z dodajanjem vlog, je potrebno uveljaviti novo konfiguracijo vlog znotraj gruče in jo ponovno zagnati. To naredimo tako, da se vrnemo na domačo stran storitve Cloudera Manager in kliknemo na modro ikono s puščico, ki se pojavi poleg storitev, katerih konfiguracijo smo spremenili oz. v spustnem seznamu poleg imena gruče izberemo možnost **Deploy Client Configuration**.

Ker je nadzorno vozlišče glede količine delovnega pomnilnika in števila virtualnih procesorjev najmočnejše, smo mu nastavili največ vlog, med njimi tudi vlogi *JobTracker* in *NameNode*. Privzeto so mu dodeljene vloge znotraj storitve **Cloudera Management Service**, ki zbira informacije o aktivnostih vozlišč, dogodkov v povezavi s Hadoopom, proži alarme, generira poročila in ostalo. Pri dodajanju vlog storitev na vsako od vozlišč dodamo vlogo *DataNode* znotraj storitve **HDFS** in vlogo *TaskTracker* znotraj storitve **MapReduce**. Slednje je potrebno, če želimo, da vsa vozlišča v gruči sodelujejo v poslih MapReduce.

Konfiguracija vlog na vozliščih je prepuščena uporabniku in njegovi presoji. V kolikor pride do napak ali opozoril v zvezi z vozlišči in njihovimi vlogami, Cloudera Manager opozori uporabnika o nepravilnostih. Ko smo na vseh vozliščih gruče dodali vloge, smo se lotili poganjanja poslov MapReduce. Slednje je opisano v naslednjem poglavju.

Poglavje 4

Poganjanje poslov MapReduce

Ko smo vzpostavili gručo, smo s poganjanjem poslov preverili njegovo delovanje. Za testiranje poganjanja poslov MapReduce smo uporabili primere s spleta [31], [33]. Platforma Hadoop sloni na programskem jeziku Java, zato je prvi prikazan primer spisan v Javi. Razvijalci, ki so ustvarili porazdeljeni sistem Hadoop, so poskrbeli tudi za tiste, ki so bolj domači v drugih programskih jezikih. Ustvarili so namreč orodje z imenom Hadoop Streaming, ki omogoča poganjanje izvršljivih datotek in skript, ki vsebujejo funkciji *map* ali *reduce*. Tako je v nadaljevanju opisan primer posla MapReduce, ki je napisan v programskem jeziku Python, opisana pa so tudi navodila za izvajanje le-tega s pomočjo orodja Hadoop Streaming.

Oba primera, prvi napisan v Javi in drugi v Pythonu, opravljata enako nalogo – gre za štetje pojavitev vsake od besed, ki se pojavi v daljšem besedilu. Oba zgleda algoritmov sta vzeta s spleta.

4.1 Primer štetja besed v programskem jeziku Java

Pri pisanju poslov MapReduce v programskem jeziku Java, v katerem je razvit Hadoop, se predstavljeni primer poslužuje programskih knjižnic, njenih razredov in vmesnikov, ki so del platforme Hadoop. Vhodne podatke, ki se

hranijo na vozliščih v obliki blokov, Hadoop s pomočjo vmesnikov pripravi v ustrezno obliko in jih nato posreduje funkciji *map*, ki vrne vmesne rezultate v obliki parov *<ključ, vrednost>*. Vmesni pari so nato obdelani v funkciji reduce, ki vrne rezulat – prav tako v parih *<ključ, vrednost>*.

1. Na spletni strani `http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#Example3A+WordCount+v2.0` se nahaja izvorna koda primera. Izvorno kodo smo shranili v datoteko **WordCount.java**.

```
1  package org.myorg;
2
3  import java.io.IOException;
4  import java.util.*;
5
6  import org.apache.hadoop.fs.Path;
7  import org.apache.hadoop.conf.*;
8  import org.apache.hadoop.io.*;
9  import org.apache.hadoop.mapred.*;
10 import org.apache.hadoop.util.*;
11
12 public class WordCount {
13  public static class Map extends MapReduceBase implements
14           Mapper<LongWritable, Text, Text, IntWritable> {
15   private final static IntWritable one = new IntWritable(1);
16   private Text word = new Text();
17
18   public void map(LongWritable key, Text value,
19           OutputCollector<Text, IntWritable>
20           output, Reporter reporter) throws IOException {
21    String line = value.toString();
22    StringTokenizer tokenizer = new StringTokenizer(line);
23
24    while (tokenizer.hasMoreTokens()) {
25      word.set(tokenizer.nextToken());
26      output.collect(word, one);
27    }
28   }
29  }
```

```
30
31  public static class Reduce extends MapReduceBase implements
32            Reducer<Text, IntWritable, Text, IntWritable> {
33   public void reduce(Text key, Iterator<IntWritable> values,
34       OutputCollector<Text, IntWritable> output,
35       Reporter reporter) throws IOException {
36     int sum = 0;
37     while (values.hasNext()) {
38       sum += values.next().get();
39     }
40     output.collect(key, new IntWritable(sum));
41   }
42  }
43
44  public static void main(String[] args) throws Exception {
45    JobConf conf = new JobConf(WordCount.class);
46    conf.setJobName("wordcount");
47
48    conf.setOutputKeyClass(Text.class);
49    conf.setOutputValueClass(IntWritable.class);
50
51    conf.setMapperClass(Map.class);
52    conf.setCombinerClass(Reduce.class);
53    conf.setReducerClass(Reduce.class);
54
55    conf.setInputFormat(TextInputFormat.class);
56    conf.setOutputFormat(TextOutputFormat.class);
57
58    FileInputFormat.setInputPaths(conf, new Path(args[0]));
59    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
60
61    JobClient.runJob(conf);
62  }
63 }
```

2. Na nadzornem vozlišču, ki poganja virtualno napravo Quickstart, ustvarimo mapo na lokaciji **/home/cloudera/hadoopExamples**, kamor shranimo datoteko z daljšim besedilom **input.txt**. Prav tako v isto mapo shranimo datoteko z izvorno kodo – **Wordcount.java**.

3. Virtualna naprava Quickstart Cloudera ima vnaprej pripravljeno vse, kar potrebuje uporabnik za takojšnjo uporabo sistema, med drugim tudi uporabniške račune, s katerimi se prijavimo v operacijski sistem CentOS. Uporabnik dostopa do operacijskega sistema kot uporabnik **cloudera**, ki je privzeti uporabniški račun. Poleg njega je registriran tudi uporabniški račun **hdfs**. Za ustvarjanje datotek v porazdeljenem datotečnem sistemu uporabimo slednjega. Tako uporabnika **cloudera** zamenjamo z uporabnikom **hdfs** in v datotečnem sistemu HDFS ustvarimo mapo **/user/cloudera** ter ji za lastnika določimo uporabnika **cloudera**. Nato kot uporabnik **cloudera** znotraj mape **/user/cloudera** ustvarimo mapo **input**. Na koncu v mapo **input** prenesemo datoteko **input.txt** z vhodnimi podatki [32].

```
sudo su hdfs
hadoop fs -mkdir /user/cloudera/
hadoop fs -chown cloudera /user/cloudera/
exit
hadoop fs -mkdir /user/cloudera/input/
hadoop fs -put /home/cloudera/hadoopExamples/input.txt
  /user/cloudera/input/
```

4. Nato prevedemo izvorno kodo datoteke **WordCount.java**. Prevedene datoteke shranimo v vnaprej pripravljeno mapo **build** [32]. Za prevajanje datoteke potrebujemo dve datoteki s končnico **jar**, in sicer **/usr/lib/hadoop/hadoop-common-*.jar** in **/usr/lib/hadoop/client/hadoop-mapreduce-client-core-*.jar**. Mesto, označeno s

simbolom *, označuje različico paketa, ki ga uporabimo za prevajanje. Tako je ukaz za prevajanje datoteke z izvorno kodo odvisen od različic nameščenih paketov v **/usr/lib/hadoop/** in **/usr/lib/hadoop-client**. Za naš primer smo pognali naslednji ukaz:

```
javac -classpath /usr/lib/hadoop/hadoop-common-2.6.0
-cdh5.4.5.jar:/usr/lib/hadoop/client/hadoop-mapreduce
-client-core-2.6.0-cdh5.4.5.jar
  /home/cloudera/hadoopExamples /WordCount.java
  -d /home/cloudera/hadoopExamples/build/ -Xlint
```

5. Ko se izvorna koda prevede, ustvarimo izvršljivo datoteko [32].

```
jar -cvf wordcount.jar
  -C /home/cloudera/hadoopExamples/build
  /home/cloudera/hadoopExamples
```

6. V tem koraku je vse pripravljeno za izvedbo posla MapReduce. To storimo z ukazom, ki mu kot parametre podamo izvršljivo datoteko in lokacijo vhodnih ter izhodnih datotek v datotečnem sistemu HDFS.

```
hadoop jar wordcount.jar org.myorg.WordCount
  /user/cloudera/input
  /user/cloudera/output
```

Ko se posel konča, lahko v mapi **/user/cloudera/output** preverimo rezultate posla. Pričakovani rezultat je izpis vsake od besed, ki nastopa v datoteki z vhodnimi podatki, in izpis njenih pojavitev.
`hadoop fs -cat /user/cloudera/output/*`
Primer izpisa rezultata:

```
Ringhorn 6
Ringhorn,  5
Ringkiobing 2
Ringland 2
Ringland. 6
```

4.2 Primer štetja besed v programskem jeziku Python

V uvodu tega poglavja je omenjeno, da platforma Hadoop omogoča poganjanje izvorne kode, ki ni napisana le v programskem jeziku Java, temveč podpira tudi izvajanje skript in programov, napisanih v programskih jezikih, kot sta Python in C++. V dokumentaciji Apache Hadoop je zapisano, da je s pomočjo Hadoop Streaming moč poganjati izvršljive datoteke in skripte, napisane v poljubnem programskem jeziku [35].

Pri uporabi orodja Hadoop Straming so vhodni podatki pretvorjeni v vrstice in posredovani na standardni vhod. Funkciji *map* in *reduce* bereta s standardnega vhoda, podatke obdelata in jih izpišeta na standardni izhod. Hadoop Streaming ustvari posel MapReduce, ga preda gruči in nadzoruje njegov potek dokler se ne zaključi.

Pri funkcijah *map* in *reduce* je pomemben format izpisa na standardni izhod. Obe izpišeta pare <ključ, vrednost>, pri čemer je ključ ločen od vrednosti s tabulatorjem. Če posel ne izpiše tabulatorja, se celotna vrstica smatra kot ključ [35].

V sklopu navodil za poganjanje poslov MapReduce je predstavljeno poganjanje programov, napisanih v programskem jeziku Python.

1. Na spletni strani `http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/` se nahaja izvorna koda primera. Izvorno kodo shranimo v datoteki **mapper.py** in **reducer.py**. Obe datoteki premaknemo v mapo **/home/cloudera/pythonHadoopExamples**.
   V isto mapo shranimo še datoteko z besedilom, katerega pojavitve besed bo posel MapReduce prešteval. Datoteko poimenujemo **input.txt**.

Programska koda skripte **mapper.py**:

```
#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()

    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

Programska koda skripte **reducer.py**:

```
#!/usr/bin/env python

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue
    # this IF-switch only works because Hadoop sorts map
    output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word
# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

2. V datotečnem sistemu HDFS ustvarimo mapo za vhodne datoteke **pythonHadoopInput** in ji dodelimo lastništvo ter vanjo prenesemo datoteko z vhodnimi podatki.

   *Če v datotečnem sistemu HDFS ni ustvarjene mape **/user/cloudera** z nastavljenim lastništvom na uporabnika **cloudera**, to storimo po navodilih iz razdelka 4.1.*

   ```
   hadoop fs -mkdir /user/cloudera/pythonHadoopInput
   hadoop fs -put /home/cloudera/pythonHadoop/input.txt
     /user/cloudera/hadoop_python_input
   ```

3. Ko vse potrebno prenesemo v datotečni sistem HDFS in pripravimo skripti s funkcijama **map** in **reduce**, poženemo posel MapReduce. Ukaz je sestavljen iz stikal *-mapper* in *-reducer*, ki kažeta na skripto **mapper.py** in **reducer.py**. V ukaz prek stikal *-input* in *-output* podamo poti do mest v HDFS, kjer hranimo vhodne in izhodne datoteke. Datotečni sistem HDFS pri tem samodejno ustvaril mapo za izhodne datoteke. Ukazu smo dodali še stikali *-file*, ki kažeta na skripti **mapper.py** in **reducer.py**. V primeru, da smo skripti *mapper* in *reducer* predhodno prenesli v porazdeljeni sistem HDFS, stikal *-file* ni potrebno vnašati, saj je pot do skript na porazdeljenem sistemu znotraj stikal *-mapper* in *-reducer* sistemu dovolj.

   ```
   sudo hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/
   streaming/hadoop-streaming-2.6.0-mr1-cdh5.4.5.jar
   -mapper /home/cloudera/pythonHadoop/mapper.py
   -file /home/cloudera/pythonHadoop/mapper.py
   -reducer /home/cloudera/pythonHadoop/reducer.py
   -file /home/cloudera/pythonHadoop/reducer.py
   -input /user/cloudera/hadoop_python_input/inputFile.txt
   -output /user/cloudera/hadoop_python_output
   ```

4. Ko se posel konča, preverimo rezultate posla v mapi, ki smo jo določili s stikalom *-output*. Rezultate izpišemo z naslednjim ukazom:

```
hadoop fs -cat /user/cloudera/hadoop_python_output/*
```

Primer izpisa rezultata:

```
Detroidward. 1
Dettermain  21
Dettermain,2
Dettingen; 5
Detzloff; 1
Deuca'lion 2
```

4.3 Statistika posla MapReduce

Vse podrobnosti v zvezi s posli MapReduce si je moč ogledati prek uporabniškega vmesnika, ki ga ponuja Hadoop. Do vmesnika dostopamo preko brskalnika. V našem primeru je to `quickstart.cloudera:50030/jobtracker.jsp`. Podatke o poslu hrani storitev *JobTracker*, ki v našem primeru teče na vozlišču **quickstart.cloudera**. Spletni naslov storitve je odvisen od imena vozlišča, ki gosti storitev JobTracker. Do uporabniškega vmesnika lahko dostopamo tudi prek povezave, če na domači strani Cloudera Manager izberemo storitev MapReduce in v vrstičnem meniju izberemo **JobTracker Web UI**.

Do statistike končanih poslov MapReduce lahko dostopamo tako, da na dnu strani vmesnika kliknemo na hiperpovezavo **JobTracker History**. Na naslednji strani se izpišejo vsi končani posli. S klikom na želenega se prikaže statistika posla. Na strani, ki se odpre, je prek hiperpovezav moč preveriti podatke o opravilih *map* in *reduce* – čas začetka in konca opravila, vozlišče, na katerem se je opravilo odvijalo, identifikator opravila in ostalo.

Poglavje 5

Primer pisanja posla MapReduce

V okviru diplomske naloge smo pripravili tudi kratek demonstracijski primer posla MapReduce, ki izračuna standardni odklon poljubne množice števil, zapisanih v tekstovno datoteko. Implementirali smo ga v programskem jeziku Python z uporabo orodja Hadoop Streaming.

5.1 Zasnova rešitve

Standardni odklon (σ, sigma) je statistični kazalec, največkrat uporabljen za merjenje razpršenosti vrednosti. Z njim je moč izmeriti, kako razpršene so vrednosti, vsebovane v populaciji. Standardni odklon je definiran kot kvadratni koren variance, s čimer je v vsakem primeru dosežena pozitivna vrednost kazalca [29].

Varianco smo za naš primer izračunali po naslednji formuli [30]:

$$\sigma^2 = (\bar{x^2}) - \bar{x}^2 = \frac{\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2/N}{N}$$

Izračun problema smo razdelili na dva dela, in sicer na del, ki ga obravnavamo znotraj funkcije *map*, in del, ki ga obravnavamo znotraj funkcije *reduce*. Glede na to, da se začetna množica vhodnih podatkov v sklopu posla

MapReduce razdeli na podmnožice, ki se v fazi *map* obdelajo vzporedno in porazdeljeno, smo v njej opravili le tiste operacije, ki se lahko izvedejo neodvisno od elementov drugih podmnožic. Tako smo v funkciji *map* prešteli število vseh elementov podmnožice, ki jo zastopajo, vsoto vrednosti elementov in vsoto njihovih kvadratov. Ko se vsa opravila *map*, ki tečejo vzporedno in porazdeljeno, končajo, dobimo vmesne rezultate. Vmesni rezultati so nato posredovani funkciji *reduce*, ki jih zbere in izračuna število vseh elementov začetne množice, vsoto njihovih vrednosti, kot tudi vsoto kvadratov. Tako smo pripravili vse potrebne podatke za izračun variance, s pomočjo katere izračunamo končni rezultat – standardni odklon množice števil.

5.2 Skripta mapper

Pri reševanju problema smo za vhodne podatke pripravili datoteko, v kateri so s presledki ločena števila, ki predstavljajo statistično populacijo. Datoteko smo v skripti **mapper** prebrali po vrsticah in pri tem računali vsoto vrednosti elementov, vsoto njihovih kvadratov in število vseh elementov. Glede na to, da so vhodni podatki shranjeni v tekstovni datoteki, so števila predstavljena z nizi, zato za vsak niz preverimo, ali gre za število. V kolikor niz ne predstavlja števila, ga preskočimo.

```
1  #!/usr/bin/env python
2  import sys
3
4  """
5  Program bere s standardnega vhoda vrstico po vrstico.
6  Za vsak niz v vrstici preveri, ali niz predstavlja stevilo.
7  V kolikor gre za stevilo, ga pristeje vsoti stevil, izracuna
8  kvadrat stevila in ga pristeje vsoti kvadratov stevil ter pri
9  steje, koliko je vseh stevil.
10
11 Program izpise koliko je vseh stevil, kaksna je njihova vsota
12 in vsota njihovih kvadratov.
13 """
```

```
14
15 result = {}
16 sum_el = 0
17 sum_sq = 0
18 n = 0
19
20 for line in sys.stdin:
21     line = line.split()
22     for num in line:
23         try:
24             if float(num):
25                 sum_el += float(num)
26                 sum_sq += pow(float(num), 2)
27                 n += 1
28         except Exception as e:
29             continue
30
31 print('%s\t%s\t%s' % (n, sum_el, sum_sq))
```

5.3 Skripta reducer

Program s standardnega vhoda bere vrstico po vrstico. Na standardni vhod prejme vmesne rezultate, ki so se izračunali v okviru opravil *map*. Opravila *map* vrnejo tri vrednosti, in sicer število elementov množice, ki so jo obdelali, njihovo vsoto in vsoto kvadratov števil. Ker se vsako od opravil *map* izvaja le na segmentu celotne množice, vrnejo delne vsote elementov in delne vsote njihovih kvadratov. Funkcija *reduce* združi vse delne vsote elementov, vsote njihovih kvadratov in število vseh elementov ter izračuna standardni odklon.

```
1 #!/usr/bin/env python
2
3 import sys
4 from math import sqrt
5
6 """
```

```
Program bere s standardnega vhoda vrstico po vrstico.
Na standardni vhod dobi podatke, ki jih pripravijo opravila map.
Iz podatkov, ki jih prebere izracuna stevilo elementov mnozice,
njihovo vsoto in vsoto njihovih kvadratov. Nato izracuna
varianco in standardni odklon.

Program izpise stevilo elementov, varianco in standardni odklon
"""

sum_el = 0
sum_sq = 0
data = []
N = 0

for line in sys.stdin:
    n, x, x_sq = line.split()
    sum_el += float(x)
    sum_sq += float(x_sq)
    N += float(n)

variance = (sum_sq - pow(sum_el, 2) / N) / N
standard_deviation = sqrt(variance)

print("N is: %d." % N)
print("Variance is: %f." % variance)
print("Standard deviation is: %f." % standard_deviation)
```

Pri poganjanju posla MapReduce za dani algoritem je potrebno sistemu eksplicitno navesti, da se v okviru posla izvede **le eno** opravilo *reduce*. V kolikor bi se izvedlo več opravil *reduce*, bi bil rezultat posla napačen, saj algoritem predpostavlja, da se vsi vmesni rezultati, ki jih ustvarijo opravila *map*, združijo in obdelajo v enem opravilu *reduce*. Če bi se izvedlo več opravil *reduce*, le-ta ne bi računala standardnega odklona vseh elementov množice, ampak več podmnožic nje. Dobili bi dva ali več končnih rezultatov, ki pa ne bi bili pravilni, saj bi dobili standardne odklone delov začetne množice, namesto enega, ki bi zaobjel celotno množico.

Argumentu, ki ga podamo ukazu za poganjanje posla MapRedce, dodamo naslednji argument.

```
-D mapred.reduce.tasks = 1
```

Poglavje 6

Poskusi

6.1 Strojna in programska oprema

Kot pove že ime samo, je bistvo porazdeljenega sistema Hadoop vzporedno in porazdeljeno procesiranje. Slednje pomeni, da za delo s tovrstnim sistemom potrebujemo množico naprav. V diplomskem delu smo predstavili rešitev, kako s pomočjo virtualnih naprav simulirati tovrsten sistem. Pri delu smo uporabili dva prenosna računalnika. Prvi, ki je poleg nadzornega vozlišča poganjal še dve delovni, je imel vgrajen dvojedrni procesor s štirimi logičnimi nitmi in 16 GB delovnega pomnilnika. Drugi računalnik je imel na voljo dvojedrni procesor in 4 GB delovnega pomnilnika ter je poganjal eno delovno vozlišče.

6.1.1 Virtualno okolje

S spletne strani podjetja Cloudera lahko uporabniki prenesejo virtualne naprave, ki jih je moč poganjati v treh različnih virtualizacijskih okoljih, in sicer KVM, VMware in VirtualBox.

Za vzpostavitev virtualnega sistema smo uporabili odprtokodno virtualizacijsko okolje VirtualBox, ki je izdelek podjetja Oracle. VirtualBox je virtualizacijska programska oprema, ki jo je mogoče namestiti na računalnike s procesorji AMD in Intel, ne glede na to, ali je na njih nameščen operacijski

sistem Windows, Linux, Solaris ali Mac. Namen programske opreme VirtualBox je razširiti osnovne zmožnosti računalnika v smeri, da lahko ta poganja različne vrste operacijskih sistemov, vzporedno z nameščenim operacijskim sistemom in ostalo programsko opremo, ki je na njem nameščena [36].

Virtualizacija daje uporabniku veliko svobode in koristi, saj zelo poenostavi nameščanje operacijskih sistemov, uporabniku omogoča vzporedno poganjanje več različnih virtualnih naprav, na katerih so lahko nameščeni povsem različni operacijski sistemi. Virtualizacijsko okolje je zelo primerno za testiranje in eksperimentiranje s sistemi, saj so le-ti ločeni od operacijskega sistema na gostitelju, zato stvari, ki se odvijajo na njih, ne vplivajo na gostitelja neposredno. Z drugimi besedami rečeno, na virtualnih napravah lahko preizkušamo, eksperimentiramo in ustvarjamo, ne da bi nas skrbelo, da bi s tem vplivali na podatke ali operacijski sistem gostitelja. Po vrhu vsega VirtualBox omogoča tudi shranjevanje trenutnih stanj sistema (ang. snapshot), kar pomeni, da lahko v primeru, ko v času testiranja ali eksperimentiranja pridemo do točke, ko se naš sistem začne obnašati drugače od pričakovanega in ga ne znamo na hitro popraviti, sistem enostavno vrnemo v zadnje shranjeno stanje, preden se nam je neprijetnost pripetila, in tako nadaljujemo z delom s točke, ko je sistem deloval po naših željah in pričakovanjih.

6.1.2 Virtualna naprava (virtual appliance)

Pri vzpostavitvi delovnih vozlišč smo uporabili vnaprej pripravljeno virtualno napravo (ang. **virtual appliance**). Gre za virtualno napravo, ki jo uvozimo v virtualno okolje in ima že nameščen operacijski sistema in določeno programsko opremo. V našem primeru smo za delo uporabili virtualno napravo z nameščeno minimalizirano različico operacijskega sistema Linux Centos 6.4. S tem smo poenostavili vzpostavitev vozlišča in skrajšali trajanje namestitve, saj nam ni bilo potrebno ročno nameščati operacijskega sistema.

Virtualno napravo z minimalizirano verzijo operacijskega sistema smo izbrali zato, ker smo želeli vzpostaviti vozlišča le s programsko opremo, ki jo

potrebujemo za delovanje vozlišča. Tako na primer delovna vozlišča nimajo nameščenega grafičnega vmesnika in ostalih komponent, ki bi po nepotrebnem zasedali vire virtualne naprave in jih za delovanje v sklopu gruče ne potrebujemo.

6.1.3 Operacijski sistemi

Podjetje Cloudera ponuja že vnaprej pripravljeno virtualno napravo, ki jo je moč poganjati znotraj virtualnega okolja VirtualBox. Virtualna naprava ima nameščen operacijski sistem in vse ostale programske pakete ter aplikacije, ki jih potrebujemo za svoje delo. Platforma teče na operacijskem sistemu Linux CentOS 6.4. Ravno slednje je bil razlog, da smo pri vzpostavitvi ostalih virtualnih vozlišč uporabili isti operacijski sistem, saj smo pri konfiguraciji le-teh lahko uporabljali enake sistemske ukaze in urejali enake konfiguracijske datoteke. Za potrebe diplomskega dela smo izbrali le enega od operacijskih sistemov, ki jih podpirata CDH 5 in Cloudera Manager 5. CDH poleg CentOS podpira tudi distribucije Red Hat, Oracle Linux, SLES, Ubuntu in Debian, ki bi jih lahko izbrali za vzpostavitev porazdeljenega sistema.

6.2 Analiza poslov MapReduce glede na konfiguracijo gruče

V tem poglavju je predstavljeno, kako se skalirajo izvedbe poslov MapReduce glede na število vozlišč, vključenih v gručo. Glede na lastnosti porazdeljenega sistema Hadoop bi moralo biti trajanje izvedbe poslov obratno sorazmerno s številom vozlišč v gruči, kar pomeni, da več kot je v gručo vpetih vozlišč, krajše je izvajanje posla MapReduce.

6.3 Vhodni podatki

Za vhodne podatke smo izbrali en GB veliko tekstovno datoteko. V poslu MapReduce smo prešteli pojavitve vsake od besed znotraj tekstovne datoteke. Datoteka predstavlja zbirko angleških besedil, ki je nastala pod okriljem projekta Gutenberg [28]. Razlog, da smo izbrali tako veliko datoteko, izhaja iz demonstracijskega stališča, saj če posel MapReduce poženemo nad datotekami manjše velikosti, ne bomo dobili reprezentativnih rezultatov. Nad podatki smo pognali algoritma, napisana v programskem jeziku Python (razdelek 4.2) in Java (razdelek 3.4).

6.4 Rezultati

1. Najprej smo pognali zahtevek na psevdogruči, saj je ta vsebovala le eno samo vozlišče. Vozlišče je imelo na voljo 8 GB delovnega pomnilnika in dve jedri virtualnega procesorja. Poganjanje posla MapReduce je potrebovalo 19 minut in 3 sekunde za algoritem v programskem jeziku Python ter 7 minut in 20 sekund za algoritem v Javi.

2. Nato smo gruči dodali vozlišče, ki smo ga poganjali na istem gostitelju. Dodatno vozlišče je imelo na voljo 2 GB delovnega pomnilnika in eno jedro virtualnega procesorja. Posel MapReduce je potreboval za delo 13 minut in 19 sekund za algoritem v jeziku Python ter 4 minute in 39 sekund za algoritem v Javi.

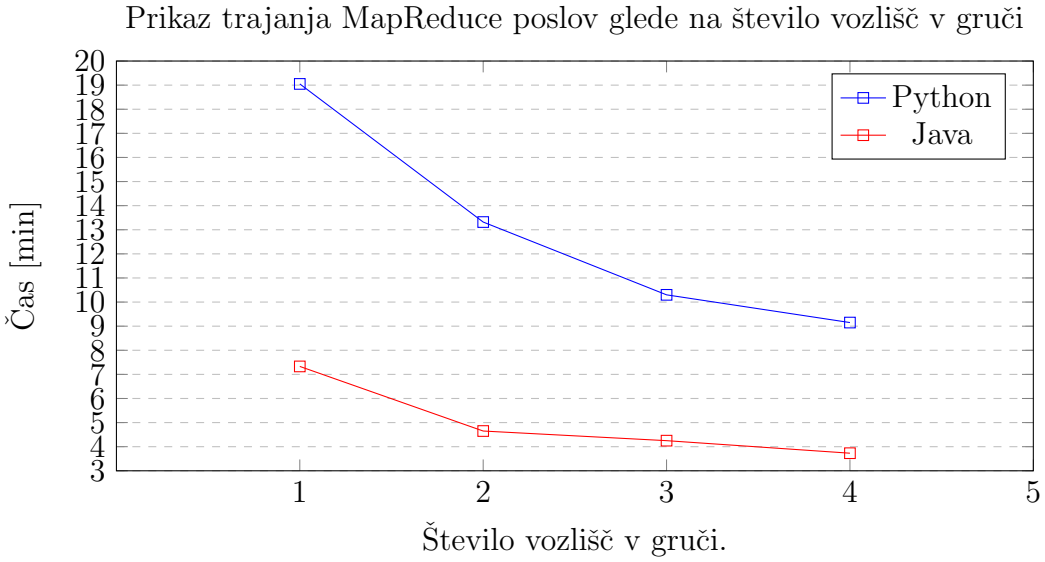
3. V tretjem koraku smo gruči dodali še eno vozlišče z istega gostitelja, ki je imelo enake strojne lastnosti kot vozlišče, navedeno v prejšnji točki. Gruča je za delo potrebovala 10 minut in 18 sekund za algoritem, napisan v jeziku Python, ter 4 minute in 15 sekund za algoritem v Javi.

4. V zadnjem koraku smo gruči dodali vozlišče, ki je bilo po specifikacijah enako vozliščema iz druge in tretje točke, le da se je to nahajalo na drugem osebnem računalniku. V zadnjem koraku je gruča za dokončanje

MapReduce zahtevka potrebovala 9 minut in 9 sekund za algoritem, napisan v jeziku Python, ter 3 minute in 44 sekund za algoritem, napisan v Javi.

Št. vozlišč	Java [t]	Python [t]	Python [t] / Java [t]
1	7 m 20 s	19 m 3 s	2,6
2	4 m 39 s	13 m 19 s	2,86
3	4 m 15 s	10 m 18 s	2,42
4	3 m 44 s	9 m 9 s	2,45

Tabela 6.1: Prikaz trajanja poslov glede na število vozlišč v gruči.



Kot je iz rezultatov razvidno, je bilo izvajanje posla MapReduce veliko počasnejše v primeru, ko smo za izvajanje posla MapReduce uporabljali orodje Hadoop Streaming in algoritem, napisan v programskem jeziku Python. Znano je, da je izvedba poslov s pomočjo Hadoop Streaming v primerih, ko gre za obdelavo velike množice podatkov, počasnejša, kot če bi

posle napisali v programskem jeziku Java, na katerem temelji Hadoop sam [34].

K počasnejši izvedbi posla, napisanega v jeziku Python, je botroval tudi sam algoritem, ki bi lahko bil napisan bolj optimalno. Vsekakor bi k boljši izvedbi pripomogla raba generatorjev in iteratorjev, kar je omenil tudi sam avtor prispevka, od koder je vzeta izvorna koda [33].

Poglavje 7

Sklepne ugotovitve

V okviru izdelave diplomske naloge smo na kratko predstavili programski model MapReduce, njegovo mesto v okviru sistema Hadoop in opisali njegovo delovanje na preprostem primeru. Osredotočili smo se na navodila in napotke za vzpostavitev ter konfiguracijo porazdeljenega sistema Hadoop s pomočjo programske rešitve CDH in vnaprej pripravljene virtualne naprave Quickstart Cloudera, ki jo nudi ameriško podjetje Cloudera Inc. V navodilih smo zajeli konfiguracijo vnaprej prilagojene virtualne naprave Cloudera Quickstart. Da bi bila simulacija porazdeljenega sistema čim bolj relevantna, smo v navodila zaobjeli tudi napotke, kako pripraviti delovna vozlišča, ki jih je moč vključiti v gručo, katere nadzorno vozlišče je Clouderina virtualna naprava.

7.1 Ugotovitve

Prepričali smo se, da lahko s storitvami, ki jih nudi Cloudera, v sorazmerno kratkem času vzpostavimo simuliran porazdeljeni sistem in da za začetek učenja in spoznavanja MapReduce potrebujemo le en boljši ali dva povprečna osebna računalnika, s katerima lahko simuliramo gručo naprav, ki poganjajo posle MapReduce. Navodila in nasveti predstavljajo odskočno desko za vse, ki se še niso srečali s sistemom Hadoop in iščejo napotke, kako vzpostaviti delovno okolje za nadaljnji razvoj. Navodila so prilagojena poga-

njanju gruče, sestavljene iz virtualnih naprav, kar daje fleksibilnost rešitvi, saj je vzpostavitev tovrstnega porazdeljenega sistema mogoča na veliki večini računalnikov, ki podpirajo virtualizacijo, ne glede na to, kateri operacijski sistem sicer uporabljajo.

Glede na to, da večina študentov Fakultete za računalništvo in informatiko v Ljubljani tekom študija le sliši za pojme Hadoop, BigData itd. in se z njimi ne sreča pobližje, menimo, da lahko to diplomsko delo služi kot nekakšen uvod v področje računalništva, ki je dandanes zelo aktualno.

7.1.1 Cloudera in Hadoop

V diplomskem delu smo predstavili postopek za vzpostavitev porazdeljenega sistema Hadoop s pomočjo programske rešitve Cloudera in uporabo virtualnih naprav. Vnaprej pripravljena virtualna naprava Quickstart Cloudera nudi vse potrebno, kar uporabnik potrebuje za takojšnje delo na platformi Hadoop. Storitvi Cloudera Manager in CDH uporabniku močno olajšata vzpostavitev porazdeljenega sistema, saj skoraj v celoti avtomatizirata proces vzpostavitve sistema. Tako je vzpostavitev sistema enostavna in hitra. Vsi koraki, ki smo jih opisali v navodilih, so bolj kot ne le nastavljanje lastnosti omrežnih vmesnikov za vzpostavitev povezave med omrežji in nastavitev nekaterih parametrov, ki so potrebni za delovanje storitev Cloudera Manager in CDH. Ker za vse detajle v povezavi s Hadoopom poskrbita Cloudera Manager in CDH, nam ni potrebno ročno nameščati potrebnih sistemskih paketov, urejati konfiguracijskih datotek in nastavljati ostalih parametrov, ki bi jih morali, v kolikor bi se odločili za konvencionalen način namestitve porazdeljenega sistema Hadoop. Pri vzpostavitvi gruče na konvencionalen način bi bilo potrebno opraviti večino enakih korakov, kot smo jih pri našem postopku, npr. nastavljanje imen vozlišč (*hostname*), urejanje datoteke *hosts*, nastavitev omrežnih vmesnikov in podobno, šele nato bi sledili vsi koraki v zvezi s sistemom Hadoop.

7.1.2 Časovna zamudnost namestitve in konfiguracije

Za vzpostavitev gruče, ki je pripravljena za uporabo in vključuje nadzorno vozlišče ter dve delovni vozlišči, pri čemer je drugo od njiju klon prvega, potrebuje povprečen uporabnik, ki ima minimalne izkušnje z delom na operacijskem sistemu Linux, slabi **dve uri**. V ta čas je vštet tudi čas nameščanja sistemskih paketov, ki so nameščeni tekom procesa. Ravno nameščanje le-teh predstavlja največji delež potrebnega časa za vzpostavitev.

Z ozirom na to, da je večina korakov nameščanja in vzpostavitve sistema Hadoop enaka tistim, ki smo jih morali izvesti pri vzpostavitvi s pomočjo storitve Cloudera Manager, lahko predvidevamo, da bi za vzpostavitev sistema na konvencionalen način porabili precej več časa, glede na to, da slednji predvideva ročno nameščanje vseh potrebnih paketov, ročno nastavitev konfiguracijskih datotek, pa tudi ročno zaganjanje in ustavljanje vseh storitev v sklopu Hadoopa, kar lahko s pomočjo Cloudera Manager in CDH opravimo le z nekaj kliki. Slednje je veliko preprosteje in hitreje.

Sistem smo med drugim vzpostavili in preizkusili tudi z različico CDH 5.5, ki je trenutno najnovejša, a se navodila za vzpostavitev bistveno ne razlikujejo, tako da je možno brez težav vzpostaviti sistem z navodili, ki smo jih predstavili v diplomskem delu, čeprav ta temeljijo na različici 5.4.8.

7.2 Možnosti za nadaljnje delo

Pri pisanju diplomskega dela ostaja še veliko prostora za izboljšave. Ena izmed teh je uporaba programske rešitve Vagrant, ki omogoča vzpostavljanje in konfiguracijo virtualnih okolij [39]. S pomočjo Vagranta bi bilo mogoče prek nastavitvenih datotek vnaprej pripraviti virtualne naprave, ki bi bile v veliki meri že pripravljene za vključitev vozlišča v gručo, ne da bi pri tem morali iti skozi večino korakov za pripravo vozlišča. To bi pripravo gruče v precejšnji meri avtomatiziralo, kar je za uporabnika, ki se hoče spoznati s sistemom Hadoop, ne da bi pri tem izgubljal čas s konfiguracijo, zelo dobrodošlo.

Dodatek

Dodatek je namenjen vsem, ki želijo računska vozlišča prilagoditi svojim željam in nanje namestiti le tisto programsko opremo, ki jo potrebujejo. Vnaprej pripravljene virtualne naprave imajo običajno nameščeno veliko programske opreme, ki pa je vozlišča, ki izvajajo striktno računske operacije, ne potrebujejo vedno. Tako lahko pripravimo vozlišča, ki za svoje delovanje potrebujejo minimalen nabor programske opreme, brez odvečnega balasta, kot je grafični vmesnik in podobno.

Vzpostavitev delovnega vozlišča brez pomoči vnaprej pripravljene virtualne naprave

1. Z uradne spletne strani Linuxove distribucije CentOS [38], prenesemo sliko ISO virtualne naprave. Izberemo verzijo 6.4 (64-bitno), glede na to, da isto uporablja tudi Quickstart Cloudera. Priporočamo, da se izbere minimalizirana verzija operacijskega sistema, saj postavljamo delovno vozlišče, kjer ne potrebujemo grafičnega vmesnika in ostalih sistemskih paketov, s čimer prispevamo k hitrejši namestitvi in uporabi virtualne naprave.

2. Odpremo program Oracle VM VirtualBox.

3. Kliknemo na moder gumb z oznako “New” v zgornjem levem kotu.

4. V vnosno polje **Name** vnesemo ime virtualne naprave, s seznama **Type**

izberemo vrsto operacijskega sistema, tj. Linux, in s seznama **Version** izberemo **Other Linux (64-bit)**.

5. Na naslednji strani izberemo količine delovnega pomnilnika. Nastavimo vsaj 2 GB delovnega pomnilnika.

6. Nato izberemo opcijo **Create a virtual hard drive now** in kliknemo gumb **Create**.

7. V naslednjem oknu izberemo prvo možnost – **VDI (VirtualBox Disk Image** in za tem vrsto virtualnega diska. Na izbiro je dinamično alociran (**Dynamically allocated**) ali fiksen (**Fixed size**) – izberemo prvega.

8. V naslednjem oknu določimo ime novega virtualnega diska in z drsnikom določimo njegovo velikost – ta naj bo 20 GB ali več.

   Po končanem 8. koraku je naša virtualna naprava pripravljena za zagon in dejansko namestitev operacijskega sistema.

9. V glavnem meniju programa VirtualBox na levi izberemo pravkar ustvarjeno virtualno napravo in pritisnemo zeleno tipko **Start**.

10. Ob zagonu virtualne naprave v pojavnem oknu izberemo lokacijo, kjer imamo shranjeno sliko ISO operacijskega sistema in pritisnemo gumb **Start**.

11. V meniju operacijskega sistema CentOS izberemo opcijo **Install or upgrade an existing system**. Zatem se bo pričela namestitev sistema.

12. Po nekaj sekundah se bo pojavilo okno, ki sprašuje, ali naj sistem pred pričetkom inštalacije preizkusi medij, na katerem je operacijski sistem. Glede na to, da operacijski sistem nameščamo s slike ISO, lahko ta korak preskočimo – izberemo opcijo **Skip**.

13. Ko se pojavi uvodno okno, kliknemo gumb **Next**.

14. Izberemo jezik namestitve.

15. Izberemo razpored tipkovnice.

16. V naslednjem meniju sistem sprašuje, kakšne vrste diskovnih naprav oz. naprav za shranjevanje podatkov bomo uporabili. Glede na to, da gre za virtualno okolje in virtualne diske, lahko izberemo opcijo **Basic Storage Devices**.

17. V kolikor se pojavi okno, kjer sistem sprašuje, ali sme zbrisati podatke na virtualnem disku, izberemo opcijo **Yes, dicard any data**, ki bo zbrisala vse podatke na virtualnem disku, a to ne bo imelo vpliva, saj smo izbrali nov virtualni disk, ki je prazen.

18. V naslednjem koraku nastavimo **hostname** oz. ime računalnika. Predlagamo, da je ime smiselno, kot je na primer **hadoop1.example.com**, saj se bomo na ime računalnika sklicevali pri povezovanju vozlišč v gruče.

19. Izberemo državo, s pomočjo katere bo sistem določil časovno cono in nastavil čas operacijskemu sistemu.

20. Nastavimo geslo za uporabnika **root**.

21. Izberemo vrsto namestitve. Predlagamo, da izberemo opcijo **Replace Existing Linux System(s)**, ki zbriše le particije Linux (od morebitnih predhodnih inštalacij) ter ne briše drugih particij, ki morebiti obstajajo na ostalih diskih.

22. V pojavnem oknu, ki opozarja, da se bodo vse izbire, ki smo jih do sedaj navedli, začele zapisovati na disk, kliknemo gumb **Write changes to disc** oz. kliknemo **Go back**, v kolikor pred namestitvijo želimo spremeniti konfiguracijo. Ko kliknemo gumb za potrditev, se bo pričelo nameščanje operacijskega sistem CentOS.

23. Ko se namestitev konča, kliknemo gumb **Reboot**, s katerim ponovno zaženemo virtualno napravo.

Navodila, kako nastaviti omrežne vmesnike, prilagoditi konfiguracijske datoteke in vse potrebno za vzpostavitev delovnega vozlišča in njegovo vključitev v gručo, so že povzeta v razdelku 3.4 od četrtega do osmega koraka in 3.5.1 od drugega koraka naprej. V kolikor želimo novo vozlišče dodati v gručo, je postopek opisan v razdelku 3.5.2.

Literatura

[1] CDH 5 Requirements and Supported Versions. [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/cdh_ig_req_supported_versions.html. [Dostopano 20. 9. 2015].

[2] R. Lämmel. Google's MapReduce programming model — Revisited, *Science of Computer Programming*, št. 70, zv. 1, str. 1–30, 2008.

[3] Hadoop MapReduce Next Generation - Cluster Setup. [Online]. Dosegljivo: https://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-common/ClusterSetup.html. [Dostopano 20. 9. 2015].

[4] L. Fortnow. Viewpoint: Time for computer science to grow up, *Communications of the ACM*, št. 52, zv. 8, str. 33–35, 2009.

[5] A Very Brief Introduction to MapReduce. [Online]. Dosegljivo: http://hci.stanford.edu/courses/cs448g/a2/files/map_reduce_tutorial.pdf. [Dostopano: 4. 8. 2015].

[6] J. Dean, S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters, *Communications of the ACM - 50th anniversary issue: 1958 - 2008*, št. 51, zv. 1, str. 107–113, 2008.

[7] Apache Hadoop NextGen MapReduce (YARN). [Online]. Dosegljivo: http://hadoop.apache.org/docs/r2.4.1/hadoop-yarn/hadoop-yarn-site/YARN.html. [Dostopano 20. 8. 2015].

[8] Funkcijsko programiranje i programski jezik Haskell. [Online]. Dosegljivo: https://www.fer.unizg.hr/_download/repository/PPIJ_Funkcijsko_programiranje_i_Haskell%5B3%5D.pdf. [Dostopano 24. 1. 2016].

[9] Map/Reduce - in Functional Programming & Parallel Processing Perspectives. [Online]. Dosegljivo: http://blogs.msdn.com/b/csliu/archive/2009/11/10/map-reduce-in-functional-programming-parallel-processing-perspectives.aspx. [Dostopano 24. 1. 2016].

[10] What is Hadoop [Online]. Dosegljivo: http://www-01.ibm.com/software/data/infosphere/hadoop/. [Dostopano 20. 8. 2015].

[11] Hadoop. [Online]. Dosegljivo: http://hadoop.apache.org. [Dostopano 22. 9. 2015].

[12] HDFS Users Guide. [Online]. Dosegljivo: http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html. [Dostopano 20. 8. 2015].

[13] Hadoop. [Online]. Dosegljivo: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. [Dostopano 20. 8. 2015].

[14] Cloudera - company profile. [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/about/company-profile.html. [Dostopano 24. 9. 2015].

[15] Introduction to networking modes. [Online]. Dosegljivo: http://www.virtualbox.org/manual/ch06.html. [Dostopano 25. 9. 2015].

[16] CDH. [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html. [Dostopano 24. 9. 2015].

[17] HADOOP: Scalable, Flexible Data Storage and Analysis. [Online]. Dosegljivo: http://www.cloudera.com/content/dam/cloudera/Resources/PDF/Olson_IQT_Quarterly_Spring_2010.pdf. [Dostopano 17. 9. 2015].

[18] Cloudera manager requirements. [Online]. Dosegljivo: http://www.cloudera.com/content/www/en-us/documentation/manager/5-0-x/Cloudera-Manager-Installation-Guide/cm5ig_cm_requirements.html. [Dostopano 20. 9. 2015].

[19] T. White (2015). *Hadoop: The Definitive Guide, 3rd Edition*, Sebastopol: O'Reilly Media, Inc., 2012.

[20] MapReduce Tutorial -Overview. [Online]. Dosegljivo: http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html. [Dostopano 20. 8. 2015].

[21] New Features in CDH 5. [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_rn_new_features.html#xd_583c10bfdbd326ba--43d5fd93-1410993f8c2--7f6d. [Dostopano: 24. 9. 2015].

[22] Cloudera Manager 5 Overview [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_intro_primer.html. [Dostopano: 24. 9. 2015].

[23] QuickStart VMs for CDH 5.4.x. [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/downloads/quickstart_vms/cdh-5-4-x.html. [Dostopano: 24. 9. 2015].

[24] Cloudera Express. [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-express.html. [Dostopano: 24. 9. 2015].

[25] Troubleshooting Installation and Upgrade Problems. [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/documentation/archives/cloudera-manager-4/v4-5-1/Cloudera-Manager-Enterprise-Edition-Installation-Guide/cmeeig_topic_19.html. [Dostopano: 12. 10. 2015].

[26] ZooKeeper. [Online]. Dosegljivo: https://wiki.apache.org/hadoop/ZooKeeper. [Dostopano: 22. 1. 2016].

[27] Optimizing Performance in CDH. [Online]. Dosegljivo: http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_admin_performance.html#xd_583c10bfdbd326ba-7dae4aa6-147c30d0933--7fd5__section_xpq_sdf_jq. [Dostopano: 12. 10. 2015].

[28] Gutenberg project. [Online]. Dosegljivo: http://pizzachili.dcc.uchile.cl/texts/nlang/. [Dostopano: 10. 12. 2015].

[29] Standardni odklon. [Online]. Dosegljivo: http://wiki.fmf.uni-lj.si/wiki/Standardni_odklon. [Dostopano: 24. 9. 2015].

[30] Varianca. [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Algorithms_for_calculating_variance. [Dostopano: 2. 1. 2016].

[31] WordCount v2.0. [Online]. Dosegljivo: `http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#Example3A+WordCount+v2.0`. [Dostopano: 14. 12. 2015].

[32] Usage. [Online]. Dosegljivo: `http://www.cloudera.com/content/www/en-us/documentation/other/tutorial/CDH5/Hadoop-Tutorial/ht_usage.html`. [Dostopano: 31. 12. 2015].

[33] Python MapReduceCode. [Online]. Dosegljivo: `http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/`. [Dostopano: 14. 12. 2015].

[34] M. Ding, L. Zheng, Y. Lu, L. Li, S. Guo, M. Guo. More convenient more overhead: the performance evaluation of Hadoop streaming, *RACS '11 Proceedings of the 2011 ACM Symposium on Research in Applied Computation, str. 307-313, 2011.*

[35] Hadoop Streaming. [Online]. Dosegljivo: `http://hadoop.apache.org/docs/r1.2.1/streaming.html`. [Dostopano: 25. 9. 2015].

[36] VirtualBox Manual. [Online]. Dosegljivo: `https://www.virtualbox.org/manual/ch01.html`. [Dostopano: 20. 9. 2015].

[37] Apache Hadoop. [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-4`. [Dostopano 20. 9. 2015].

[38] CentOS. [Online]. Dosegljivo: `http://vault.centos.org/6.4/isos/x86_64/`. [Dostopano 1. 12. 2015].

[39] Vagrant Docs. [Online]. Dosegljivo: `http://docs.vagrantup.com/v2/why-vagrant/index.html`. [Dostopano 12. 12. 2015].