1.1. Definirajte pojam velikih podataka (Big Data).

Big Data je velika količina podataka sakupljenih tijekom određenog vremenskog intervala. Takvi su podatci teški za analizu preko standardnih alata.

1.2. Navedite 4 važna obilježja velikih podataka.

Volumen (Volume), Raznolikost (Variety), Brzina (Velocity), Vjerodostojnost (Veracity)

1.3. Objasnite raznolikost kao jedno od 4 glavna obilježja velikih podataka.

Raznolikost znači da imamo različite vrste velike količine podataka koje se mogu razlikovati. Neki su od primjera statističke informacije o zdravstvenom osiguranju, Facebook ili Twitter objave, video, slika, podaci sa senzora …

1.4. Objasnite zašto Hadoop za svoj rad ne zahtijeva specijalan hardver, već radi na standardnim računalima.

Hadoop radi na standardnim računalima zato što je napravljen da lako može raditi na većem broju računala te na njima postiže visoku raspoloživost otkrivanjem i rukovanjem (očekivanih) ispada na aplikacijskom sloju.

Svako racunalo sadrzi odredjeni dio podatka i samo se za njega brine. Ideja je da je lakse skalirati horizontalno (dodavati jos racunala) nego vertikalno (nadogradjivati racunalo).

2.1. Usporedite glavne prednosti i nedostatke raspodijeljenih datotečnih sustava naspram lokalnih datotečnih sustava.

Prednosti raspodijeljenih datotecnih sustava:

- Vrijeme citanja velikih podataka, umjesto da citamo odjednom 1TB file mozemo imati tu istu datoteku raspodijeljenu na 100 diskova sa kojih citamo paralelno po 10GB sto je u konacnici puno brze, osigurana je perzistencija podataka.
- Veca sigurnost od kvarova medija

Nedostaci

- Postoji puno “overhead” prilikom citanja malih podataka zato jer podaci idu preko mreze te blokovi HDFS-a su kapacitirani za velike datoteke.
- Sporo je zapisivati i azurirati u raspodijeljene datotecne sustave (tehnički možeš samo appendati na postojeće)

2.2. Što znači pojam "lokalnosti podataka" (*data locality*) u kontekstu raspodijeljenih datotečnih sustava?

Lokalnost podataka predstavlja obradu podataka tamo gdje su i pohranjeni čime se štedi na mrežnom prometu i brzini obrade.

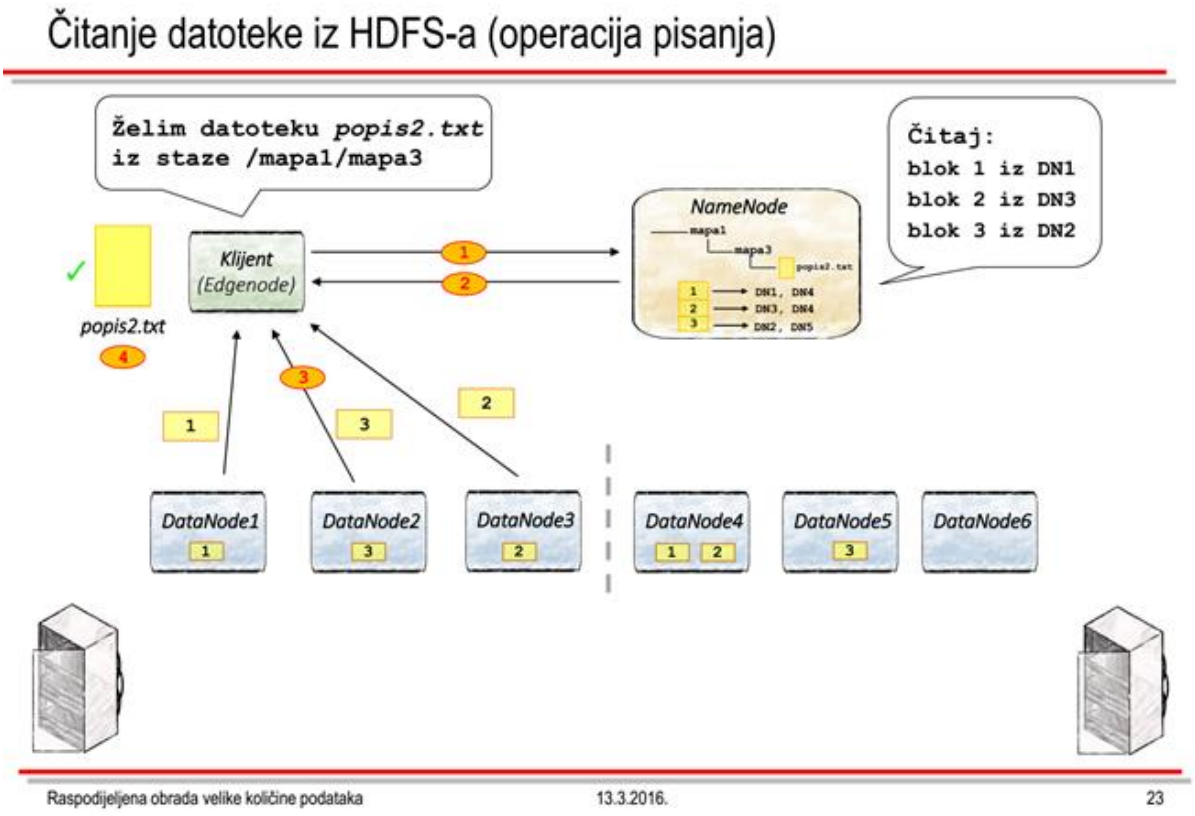
2.3. Opišite ulogu i glavne funkcionalnosti podatkovnog čvora u raspodijeljenom datotečnom sustavu.

Podatkovni čvor je radni (slave) čvor koji pohranjuje podatkovne blokove. On šalje 2 glavna tipa poruka imenskom čvoru: „heartbeat“ poruke signal da je podatkovni signal živ i „block report“ – izvještaj koje blokove čvor pohranjuje. Klijent direktno komunicira sa čvorom radi dohvata podataka.

2.4. Koja je uloga datoteka *FSImage* i *EditLog* na imenskom čvoru HDFS-a?

FSImage i EditLog su dvije glavne datoteke: FSImage čuva „snapshot“ metapodataka sustava (logička struktura, informacije o podatkovnim čvorovima, itd.), dok EditLog sadrži dnevnik izmjena nad sustavom.

2.5. Skiciraj proces čitanja datoteke iz HDFS-a.



2.6. Što rade sljedeće naredbe?

1) hdfs dfs –ls /
2) hdfs dfs –tail /user/pero/podaci.txt
3) hdfs dfs –rm /user/pero/podaci.bin

1) Vraća sadržaj mape /.

2) Pregledava redak s kraja zadane datoteke /user/pero/podaci.txt.

3) Briše datoteku podaci.bin s lokacije na HDFS-u /user/pero/podaci.bin

3.1. Objasnite što je MapReduce i navedite osnovna načela MapReducea.

MapReduce je programski model za paralelnu obradu podataka koji omogućuje analizu i obradu velikih skupova podataka u grozdu računala.

Osnovna načela su: podjela posla na manje neovisne poslove, pridjeljivanje poslova radnim procesima raspodijeljenim u velikom grozdu računala, osiguravanje pristupa podatcima potrebnim za obradu, sinkroniziranje neovisnih radnih procesa i otpornost na neispravnosti.

3.2. Opišite izvođenje MapReduce programa. Koliko poslova (*job*), a koliko zadataka (*task*) se izvrši tijekom izvođenja jednog MapReduce programa ako znamo da nije bilo neispravnosti tijekom izvođenja programa.

1. Korisnički MapReduce zadatak (job): sastoji se od ulaznih podataka, MapReduce programa i konfiguracijske datoteke
2. Okolina MapReduce za dani MapReduce program kreira kopije procesa u grozdu računala, jedan postaje master, a ostali worker procesi.
3. Master proces raspodjeljuje m map i r reduce zadataka (task) slobodnim worker procesima. → m == broj dijelova ulaznih podataka (splits), obično definira okolina MapReduce (programer može sugerirati), r == broj reduce procesa definira programer
4. Svaki map proces čita dodijeljene podatke i izvodi definiranu map funkciju. Okolina obično izvodi map proces na računalu na kojem je pohranjena kopija dodijeljenih podataka.
5. Svaki map proces po završetku obrade sortira i dijeli međurezultat na particije, broj particija ovisi o r. Također obavještava proces o lokaciji međurezultata.
6. Master proces javlja lokacije međurezultata reduce procesu koji na njima izvodi funkciju reduce i sprema perzistentno rezultate izvođenja.
7. Master proces dojavljuje korisničkom MapReduce procesu da je obrada završena. Rezultati se obrade nalaze u definiranom direktoriju.

3.3. Objasnite razliku između funkcija Combiner i Partitioner. Ako MapReduce program sadrži obje funkcije, u kojoj fazi i kojim redoslijedom se one izvode?

Partitioner je odgovoran za dijeljenje prostora ključeva međurezultata u particije i dodjeljivanje svake particije reduceru te sudjeluje u fazi “shuffle and sort”, dok se Combiner koristi u mapperu za lokalnu agregaciju međurezultata. Prvo se izvodi Combiner (odmah nakon mapa sa ciljem agregacije -> smanjenja broja međurezultata) pa onda Partitioner.

Može source gdje je navedeno da se prvo izvodi Combiner, pa Partitioner?

http://stackoverflow.com/questions/22061210/what-runs-first-the-partitioner-or-the-combiner

3.4. Analizirajte sljedeći pseudo kod i napišite koji posao obavlja. Objasnite mogu li se poboljšati performanse izvođenja uvođenjem funkcije Combiner, odnosno uvođenjem funkcije Partitioner.

```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term t ∈ doc d do
4:             Emit(term t, count 1)

1: class Reducer
2:     method Reduce(term t, counts [c1, c2, ...])
3:         sum ← 0
4:         for all count c ∈ counts [c1, c2, ...] do
5:             sum ← sum + c
6:         Emit(term t, count sum)
```

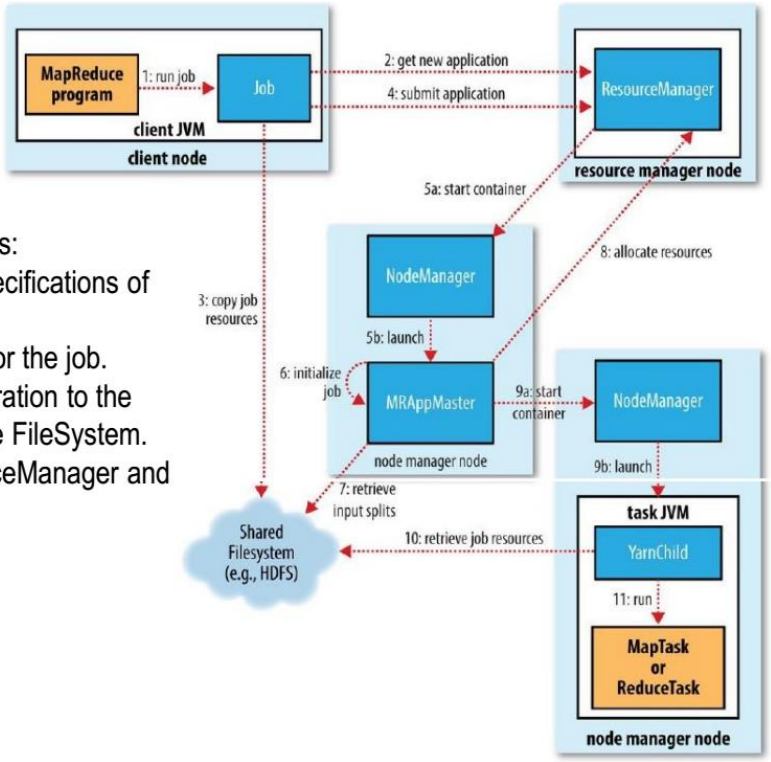
Navedeni pseudokod obavlja posao brojenja broja pojavljivanja pojedine riječi. Klasa mapper šalje pojedinu riječ s vrijednosti 1, dok reducer zbraja broj pojavljivanja iste te riječi te šalje pojedinu riječ sa sumom pojavljivanja kao vrijednosti.

Moguće je poboljšati korištenjem combinera koji uz prebrojavanje pojavljivanja riječi za pojedinačni dokument također grupira vrijednosti po ključevima čime se onda smanjuje veličina skupa međurezultata. Taj međurezultat dalje obrađuje reducer. S partitionerom ne dolazi do poboljšanja jer nemamo učinkovit način na koji bismo podijelili ključeve na particije.

4.1. Objasnite zašto Hadoop koristi vlastiti format za serijalizaciju podataka. Navedite dva najvažnija sučelja koja se koriste pri serijalizaciji podataka u Hadoopu. Objasnite u kojem slučaju moramo raditi vlastitu implementaciju ovih sučelja.

Hadoop koristi vlastiti format za serijalizaciju podataka zato što omogućuje kompaktnu, brzu i učinkovitu (de)serijalizaciju te je proširiv i interoperabilan. Dva najvažnije sučelja koja se koriste pri serijalizaciji podataka u Hadoopu su: *Writable* i *WritableComparable*. Vlastitu implementaciju ovih sučelja moramo raditi kad moramo kontrolirati binarni prikaz podataka i pravila za njihovo sortiranje.

4.2. Skicirajte i ukratko objasnite način izvođenja posla u programskom modelu MapReduce, verzija 2.0 (YARN).



The job submission process involves:
1.Checking the input and output specifications of the job.
2.Computing the InputSplit values for the job.
3. Copying the job's jar and configuration to the MapReduce system directory on the FileSystem.
4. Submitting the job to the ResourceManager and optionally monitoring it's status.

4.3. Napišite pseudokod metoda map, reduce i combine za ispravan izračun srednje vrijednosti zapisa u programskom modelu MapReduce.

```
1: class Mapper
2:     method Map(string t, integer r)
3:         Emit(string t, pair (r, 1))

1: class Combiner
2:     method Combine(string t, pairs [(s1, c1), (s2, c2) ...])
3:         sum ← 0
4:         cnt ← 0
5:         for all pair (s, c) ∈ pairs [(s1, c1), (s2, c2) ...] do
6:             sum ← sum + s
7:             cnt ← cnt + c
8:         Emit(string t, pair (sum, cnt))

1: class Reducer
2:     method Reduce(string t, pairs [(s1, c1), (s2, c2) ...])
3:         sum ← 0
4:         cnt ← 0
5:         for all pair (s, c) ∈ pairs [(s1, c1), (s2, c2) ...] do
6:             sum ← sum + s
7:             cnt ← cnt + c
8:         r_avg ← sum/cnt
9:         Emit(string t, integer r_avg)
```

4.4. Opišite načelo rada oblikovnog obrasca za filtriranje zapisa u programskom modelu MapReduce. Objasnite ima li smisla koristiti combinera u ovom oblikovnom obrascu?

Oblikovni obrazac za filtriranje pronalazi podskup podataka koji zadovoljavaju zadane kriterije. Koristi se samo funkcija map i to za filtriranje, dok se reduce može koristiti za kopiranje izlaza u jednu izlaznu datoteku. Ne isplati se koristiti combiner jer podatke filtriramo već u mapu pa njegovo korištenje nema smisla.

4.5. Opišite načelo rada oblikovnog obrasca za dijeljenje (*partitioning*) zapisa u programskom modelu MapReduce. Objasnite ima li smisla koristiti combinera u ovom oblikovnom obrascu?

Partitioning dijeli ulazni skup podataka u preddefinirani broj particija prema definiranom kriteriju. Koristi se posebni partitioner koji definira za kojeg je reducera namijenjena određena particija.

Ne isplati se koristiti combiner jer dijelimo prostor ključeva na particije i dodjeljujemo svaku particiju reduceru.

5.1. Tražilica je kao odgovor na upit vratila 70 dokumenata od kojih je 35 relevantnih od ukupno 42 relevantna dokumenta za dani upit. Pretraživana kolekcija sadrži 7000 dokumenata. Izračunajte preciznost i odziv tražilice.

- Odziv (engl. *recall*)
  - postotak relevantnih dokumenata iz odgovora u odnosu na ukupni broj relevantnih dokumenata u kolekciji

$$Recall = \frac{|A \cap R|}{|R|}$$

- Preciznost (engl. *precision*)
  - postotak relevantnih dokumenata iz odgovora u odnosu na ukupni broj dokumenata u odgovoru

$$Precision = \frac{|A \cap R|}{|A|}$$

Odziv = broj relevantnih dokumenata / broj relevantnih dokumenata u relaciji = 35/42 = 83.33%

Preciznost = broj relevantnih dokumenata / ukupni broj dokumenata vraćenih upitom (u odgovoru) = 35/70 = 50%

5.2. Objasnite način rangiranja dokumenata u vektorskom prostornom modelu.

Za rangiranje dokumenata u odgovoru na upit koristi se mjera sličnosti dokumenta i upita. Sličnost dokumenata dj i dk računa se kao kosinus kuta među njihovim vektorima.

v2: Riječi su dimenzije prostora, dokumenti i upiti su vektori tog prostora. Sličnost dokumenata i upita je proporcionalna kosinusu kuta između njihovih vektora, odnosno što je kut manji, dokumenti i upit su sličniji.

5.3. Napišite pseudo-kod za generiranje invertiranog indeksa pomoću programa MapReduce.

```
1: class Mapper
2:     procedure Map(docid n, doc d)
3:         H ← new AssociativeArray
4:         for all term t ∈ doc d do
5:             H{t} ← H{t} + 1
6:         for all term t ∈ H do
7:             Emit(term t, posting ⟨n, H{t}⟩)

1: class Reducer
2:     procedure Reduce(term t, postings [⟨n1, f1⟩, ⟨n2, f2⟩ ...])
3:         P ← new List
4:         for all posting ⟨a, f⟩ ∈ postings [⟨n1, f1⟩, ⟨n2, f2⟩ ...] do
5:             P.Add(⟨a, f⟩)
6:         P.Sort()
7:         Emit(term t, postings P)
```

5.4. Na koji način biste podijelili indeks u raspodijeljenoj web-tražilici? Objasnite.

Indeks web tražilice je povoljnije raspodijeliti po dokumentima, jer će tada svi poslužitelji obrađivati svaki upit te će opterećenost poslužitelja biti ravnomjerna. Veličina rezultata koje poslužitelji vraćaju je također manja u odnosu na raspodjelu po riječima.

5.5. Može li pojedini čvor izračunati vrijednost parametra idf (engl. *inverse document frequency*) za slučaj kada je indeks raspodijeljen prema dokumentima? Objasnite.

Za računanje idf je potrebno znati broj dokumenata u kojima se neka riječ pojavljuje. Obzirom da poslužitelj sadrži podatke samo o dijelu dokumenata, ne može izračunati idf, jer se riječ za koju se računa idf može pojavljivati u dokumentima za koje poslužitelj nije zadužen.

Zbog toga je potrebno globalne vrijednosti poput idf održavati MapReduce poslovima na razini cijele kolekcije.

5.6. Objasnite razliku između algoritama *Pairs* i *Stripes*.

Algoritam *Pairs* u mapu pronalazi parove riječi u istom kontekstu te emitira (par, 1), dok algoritam *Stripes* u mapu pobrojava susjedne riječi u posebno polje i emitira par (riječ, polje_frekvencija). Kod algoritma *Pairs* reduce pobrojava pojavljivanje istih parova riječi, dok kod algoritma *Stripes* prebrojava frekvencije pojavljivanja susjednih riječi iz polje_frekvencija za svaku riječ.

6.1. Oblikovni obrazac za pronalaženje k-najboljih zapisa je jedan od obrazaca filtriranja u programskom modelu MapReduce. U ovom oblikovnom obrascu mapper i reducer koriste metode setup() i cleanup() za inicijalizaciju liste k-najboljih i generiranje izlaznih parova vrijednosti. Napišite pseudokod metoda map i reduce. Objasnite ima li smisla koristiti combinera u ovom oblikovnom obrascu.

```
1: class MAPPER
2:         method SETUP
3:                 topk ← new TopkList
4:         method MAP(key, value)
5:                 topk.add(value)
6:                 if (top.size() > 10) then
7:                         topk.pollFirst()
8:         method CLEANUP
9:                 foreach(value in topk)
10:                        EMIT(null, value)
11: class REDUCER
12:        method SETUP
13:                topk ← new TopkList
14:        method REDUCE(key, values)
15:                foreach(value in values)
16:                        topk.add(value)
17:                        if (top.size() > 10) then
18:                                topk.pollFirst()
19:                foreach(value in topk)
20:                        EMIT(null, value)
```

Nema smisla koristiti combiner jer se rezultat na strani mapera ne može dodatno sažeti.

6.2. Opišite načelo rada oblikovnog obrasca za pronalaženje različitih zapisa u programskom modelu MapReduce. Objasnite ima li smisla koristiti combinera u ovom oblikovnom obrascu?

Oblikovni obrazac za pronalaženje različitih zapisa u programskom modelu MapReduce funkcionira tako da kao rezultat želimo dobiti jedinstvene zapise. U mapu emitiramo ključ i null vrijednost u obliku (ključ, null) za svaki zapis. Ne isplati se koristiti combiner jer su nam svi zapisi isti i zato što nam je samo ključ važan.

6.3. Oblikovni obrasci za dijeljenje (*partitioning*) zapisa i grupiranje (*binning*) zapisa u programskom modelu MapReduce se koriste za kategoriziranje zapisa. Objasnite način rada jednog i drugog obrasca. Koliko će izlaznih datoteka nastati pri korištenju obrasca za dijeljenje zapisa, a koliko korištenjem obrasca za grupiranje zapisa?

Oblikovni obrazac za dijeljenje (*partitioning*) funkcionira da tako da dijeli ulazni skup podataka u preddefinirani broj particija prema definiranom kriteriju te za to koristi posebnog partitionera koji definira za kojeg je reducera namijenjena određena particija. Koristi identity mappera i reducera koji direktno mapiraju ulaze na izlaze te prilagođenog partitionera. Cijeli posao dijeljenja ima jednu izlaznu datoteku po kategoriji.

Oblikovni obrazac za grupiranje zapisa (*binning*) funkcionira slučno dijeljenju zapisa, ali za kategorizanje koristi drugu fazu programskog modela MapReduce. Koristi mappera za kategoriziranje zapisa te nije potreban reducer zato što svaki mapper ima jednu izlaznu datoteku po kategoriji.

6.4. Opišite načelo rada oblikovnog obrasca za miješanje (shuffling) zapisa u programskom modelu MapReduce. Objasnite ima li smisla koristiti combinera u ovom oblikovnom obrascu?

Mapper zapisu dodaje nasumični ključ te reducer sortira zapise po tom nasumičnom ključu kojeg je dodao mapper. Nema smisla koristiti combiner u ovom oblikovnom obrascu zato što nam nije cilj agregirati iste ključeve prema vrijednosti, nego drugačije poredati postojeće zapise.

6.5. Objasnite praktičnu prednost 2 slijedno ulančana posla u programskom modelu MapReduce u odnosu na 2 uzastopno pozvana posla. Koje poslove se isplati slijedno ulančavati, a koje izvoditi paralelno?

Dva su slijedno ulančana posla povezana i omogućuju pozivanje drugog posla samo u slučaju da je prvi posao završio te je izlazni direktorij prvog slijedno ulančanog posla jednak ulaznom direktoriju drugog slijedno ulančanog posla. Paralelno poslove koji su nezavisni, slijedno poslove koji su zavisni, a ne mogu se svi obaviti na jednom grozdu.