LØSNINGSSKISSE TIL EKSAMENSOPPGAVE I FAG TDT4305 – MAI 2016

NB! Dette er ikkje fullstendige løysingar på oppgåvene, kun skisse med viktige element, hovudsakleg laga for at vi skal ha oversikt over arbeidsmengda på eksamen, og som huskeliste under sensurering. Det er også viktig å være klar over at det også kan vere andre svar ein dei som er gjeve i skissa som vert rekna som korrekt om ein har god grunngjeving.

Oppgåve 1

Tre viktigste V'ar:

1) *Volume*: enorme mengder data kontinuerlig generert, både brukargenererte og maskingenererte (t.d. sensorar)
   Ikkje i seg sjølv nok til at det er "Big Data"!
2) *Velocity*: hastigheit på datagenerering og prosessering, ofte i sanntid, t.d. umiddelbar deteksjon av trendar på Twitter, ofte straum-data
3) *Variety*: variasjon i typar data, kombinasjon av strukturerte og ustrukturerte

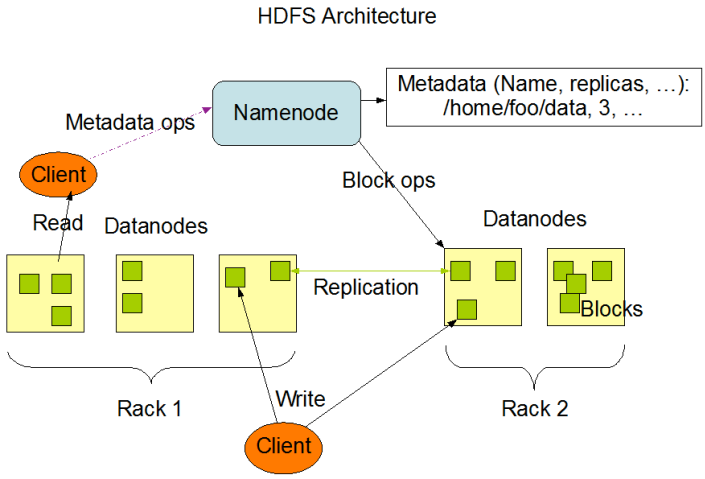
Viktig med forklaring til kvar av dei (særleg på variety er det mange som har tynn forklaring, tildels også på velocity).

Oppgåve 2

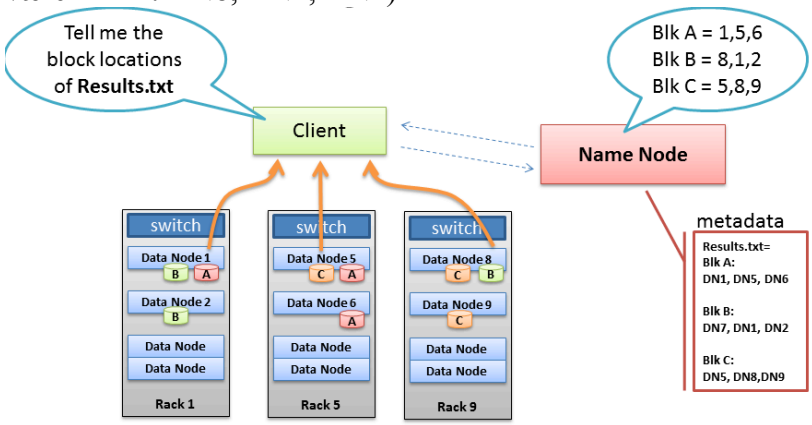
a) Mål:
   1) (Veldig) store filer
   2) Datastraum-aksess
   3) Hyllevare (maskiner)

   Ikkje eigna til:
   1) Data-aksess med krav til liten forseinking (*low-latency data access)*
   2) Mange små filer
   3) Fleire samtidige skrivarar (dvs. er *single-writer*)
   4) Vilkårlege oppdateringar i filer (dvs. er *append-only*)

b) Viktig å få med:
   1) Blokker
   2) Replikering
   3) Klient
   4) Navnenode
   5) Datanoder
   6) Aspekt rundt filer og blokker (nokre misforstår og trur granulariteten er fil og ikkje blokk)



c) Finn blokk-lokasjon(ar) (Klient kontaktar NameNode for å få liste over blokker/noder)
Les fil (blokkar) sekvensielt, velg blokk som er ”nærmast” (liten feil i figuren: på metadata skal det være Blk B: DN8, DN1, DN2)



d) Utføring av applikasjon på YARN:

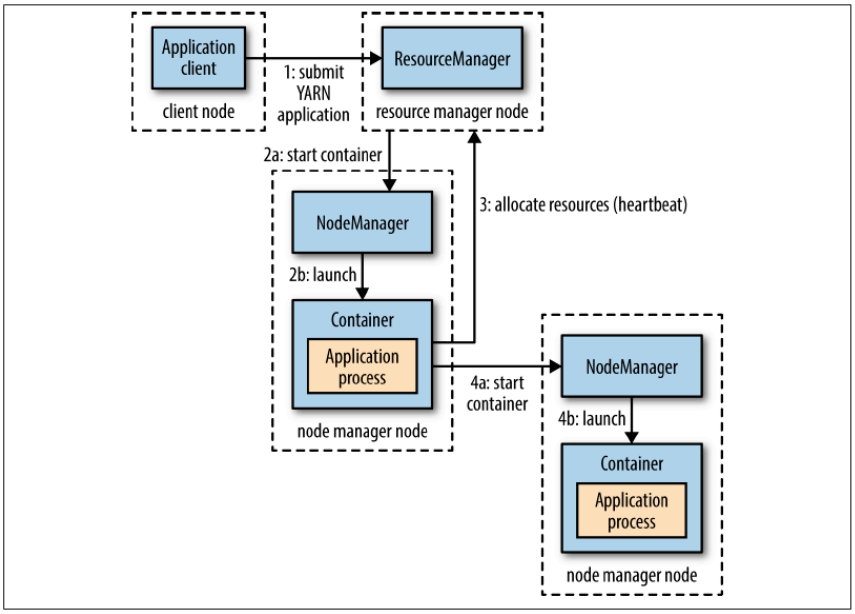


Figure 4-2. How YARN runs an application

Oppgåve 3

a) public void map(key(name), value(age,salary))
write(age,salary)

public void reduce(key, Iterable values)
n = 0
total = 0
foreach val in values
n++
total = total + val
avg = total/n
write(key, avg)

b) Utifrå oppgåveteksta kan vi anta at vi allereie har lese fila inn i ein RDD av par (key,value)=(age,salary), dvs. RDD[(int,int)]. Dette kan gjerast med dei første 3 linjene under (studentane treng altså ikkje ha med desse i svaret):

```
val lines = sc.textFile("PersonInfo.txt")
val records = lines.map(_.split("\t"))
val tuples = records.map(rec => (rec(1).toInt, rec(2).toInt))
```

Mest elegant løysing (Python og Java er også OK):

```
val maxSalary  = tuples.reduceByKey((a, b) => Math.max(a, b))
```

(Det er forventa at dei også har med funksjonen inne i reduceByKey for å vise at dei forstår korleis den vert brukt)

Mindre elegant/effektivt for eksempel:

```
val k = tuples.groupByKey()
val kk = k.mapValues(x => x.toArray.max)
```

Oppgåve 4

a) Det som er prinsippet her er å kunne utnytte sammensatte nøkler. Dvs. vi må ha en tabell med nøkkelen (ExamNo, StudentNo) gjerne sammen med all info vi trenger til å lage resultatet: (ExamNo, StudentNo, CourseName, Grade). Tilsvarende for det andre queriet: (StudentNo, CouseName, StudentName, Grade).

Student(SNo, Name, Email)
Exam(ENo, CourseName, EDay, EMonth, EYear, Duration)
ExamResult(ExamNo, StudentNo, CourseName, Grade)
StudenResult(StudentNo, CourseName, StudentName, Grade)

Også godteke StudentResult(StudentNo,ExamNo...)

b) MongoDB:

- Definerer «shard key» som brukes til å fordele dokumentene mellom tjenere. Dette må være et indeksert felt som finnes i alle dokumenter.
- Range partisjonering eller hash partisjonering
- Tag-aware sharding hvor «shards» kan plasseres i spesielle clusters som f.eks. er geografisk nær.

Hbase:

- Data lagres i «regions» som er sorterte ranges av data som lagres sammen.

- Systemet begynner med en region per tabell. Men tabellen blir splittet automatisk i flere regioner når den blir for stor. Regioner kan også automatisk blitt slått sammen når de blir for små.
- Hver region lagres på en region-server, men en region-server kan lagre mange regioner.

Oppgave 5 – Datastrømmer (streaming data) – 30 % (Alle deler teller likt)

Du skal analysere hvor mange ganger et tema om amerikansk valg og valgkamp blir nevnt i meldinger i sosiale media som Twitter.

a) Drøft karakteristikkene og/eller utfordringene med datastrøm. Nevn to andre eksempler hvor man er nødt til å håndtere en datastrøm (i tillegg til Twitter og eller sosiale media generelt).

Karakteristikk/Utfordring:
- Data kommer kontinuerlig
- Massiv
- Ubegrenset
- Muligens uendelig og derfor uhåndterlig feks. i forhold til minneforbruk.

Eksempler:
- Data fra sensorer som værstasjoner
- Søketermer fra websøkegrensesnitt
- Datastrøm generert fra klikking på URL'er.

b) Vi skiller mellom to typer spørringer når det gjelder datastrøm. Forklar hva disse er. Bruk eksempler til å støtte forklaringen din.
- **Ad-hoc-spørringer :** Normale spørringer som er kjøres en gang på strømmen.
  Eksempel: Hva er den maksimale verdien sett så langt i strømen *S*?
- **Kontinuerlig/stående søk:** Spørringer som i prinsippet kjøres på strømmen hele tiden.
  Eksempel: Rapporter hver gang en ny maksimumsverdi er sett i en strøm *S*.

c) Se for deg at du skal finne ut hvor stor andel av meldingene er relatert til temaet "valg" og "valgkamp" i en gitt begrenset tidsperiode. Til dette formålet velger vi å bruke glidende-vindu-prinsippet ("sliding window"). Anta at dette vinduet har en størrelse på 1000 twitter-meldinger (dvs. Tweets). Vis hvordan du går fram for å beregne denne andelen.

Skisse:
- Sett $N = 1000$
- For de første N meldinger regn/tell tweets som inneholder temaet "valg" og "valgkamp", og del på N.
- Deretter for hver ny melding m_i. Hvis m_i inneholder temaet legg/endre snittet med $(1 - j)/ N$, hvor m_j er den eldste meldingen, og j=1 hvis m_j inneholder temaet, 0 ellers.

d) Kan problemet over sees på som en variant av "bit counting"? Begrunn svaret ditt.

Ja, man kan sette opp en bitstrøm hvor en bit har verdien 1 dersom en melding inneholder temaet og 0 ellers. Deretter kan man telle antall enere i et gitt vindu av størrelse N, osv.

e) Bruk "bloom filter"-prinsippet til å fylle ut tabellen nedenfor

Strømelement	**Hash-funksjon - h_1**	**Hash-funksjon - h_2**	**Filtrere Innhold**
			00000000000
39 = 10 0111	011 = 3	101 = 5	00010100000
214 = 1101 0110	1110 = 14 => 3	1001 = 9	00010100010
353 = 01 0110 0001	11001 = 25 => 3	00100 = 4	00011100010

Hint: bruk *h(x)= y mod 11*, der *y* er hentet henholdsvis fra oddetalls-bits fra *x* eller partalls-bits fra *x*.

Eksempel: $h_1(39) = 011 = 3$, $h_2(39) = 101 = 5$, osv.

f) Anta at vi vil analysere de 11 siste meldingene som er kommet inn. Generelt på twitter, vil mange av meldingene bli sendt på nytt av samme bruker for å markere sitt synspunkt. Andre brukere vil "re-tweete" for å få flere til å få med seg meldingene. Forklar hvordan vi kan bruke bloom-filtre for å filtrere bort slike meldinger. Gjør de antakelsene du finner nødvendig.
   - Gjør meldingstrømmen om til bitstrøm som forklart i d)
   - Gjør alle 11 bits til 0
   - Bruk hash-funksjon som i e), og sett bit nr. *h(x)* til 1 for hver input *x*.
   - Hvis alle bits ender med 1 har vi sett strømmen før.

g) Anta nå at når de 11 meldingene har kommet inn har vi fått en strøm av data som ser ut som dette: 10100101010. Kan vi ha sett meldingen som kan representeres ved y = 1111011 før? Begrunn svaret ditt.

Her bruker vi samme prinsippet som i e). $h_1(y) = 1101 = 2$, $h_2(y) = 111 = 7$. I meldingen er bit nr. 2 1 og bit nr. 7 er 1, derfor kan vi ha sett strømmen før.

Oppgave 6 – Anbefalingssystem (recommender systems) – 20 % (6% på a.i, 4% på a.ii, 10% på b)

Du er nyansatt i et nytt firma som vil spesialisere seg på strømming av film. En av oppgavene dine er å utvikle gode anbefalingsalgoritmer og metoder.

a) En del av metoden du foreslår går ut på å gi brukeren mulighet til å "rate" filmene for så bruke dette til å finne ut hvilke filmer systemet deres skal anbefale senere. Anta at brukerne deres har "rated" følgende 10 filmer med 3 eller flere stjerner:

   Jurasic Park (Fantasi/SciFi), Harry Potter (Fantasi/Adventure), ET (SciFi), Lord of the Rings (Fantasi/Adventure), Alien (SciFi), Terminator (SciFi), 101 Dalmatians (Adventure/Family), Titanic (Romantic), Sleepless in Seattle (Romantic) og Mr. Bean (Comedy).

   i. Forklar hvordan du vil gå fram for å anbefale neste film til denne brukeren. Gjør de antakelsene du finner nødvendig.
   Finne alle filmer som er mest lik disse til å anbefale neste. Dette kan gjøres ved å finne først hvordan bruker har rated dem, og deretter estimere rating for en ny film basert på en likhetsbergning.

   ii. Ville du brukt innholdsbasert ("content-based") anbefalingsmetode eller "collaborative filtering"? Begrunn svaret ditt.
   Siden selskapet er nytt, har man ikke mange nok brukergrunnlag til å bruke collaborative filtering (for eksempel pga. sparsity). Vi bør derfor bruke content-baset anbefaling hvor vi finner og anbefaler film(er) som er mest lik(e) disse. Her kan vi bruke alle filmattributtene (som sjanger, skuespillere osv.) som grunnlag (for eksempel som features).

b) Anta følgende brukerratingstabell.

movies \ users	1	2	3	4	5	6	7	8
1	1		3			5		
2			5	4			4	
3	2	4		1	2		3	
4		2	4		5			4
5			4	3	4	2		
6	1		3		3		A	2

Bruk "***item-item collaborative filtering***"-metoden til å foreslå bruker nr. 7 sin rating av film nr. 6. Dvs. hva blir ratingverdien A? Du må vise mellomregningen.

Til denne oppgaven vil du trenge følgende formler:

Pearson Correlation similarity - likhet mellom vektor x, og vektor y:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

der r_{xs} er bruker s sin rating på film x og $\overline{r}_x$ (overline) er gjennomsnitt av alle ratingene på film x.

Vektet gjennomsnitt (weighted average) for en brukers ratinger:

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

r_{ix} er her bruker x sin rating på film i, mens s_{ij} er likhet (similarity) mellom rating-ene til film i og j.

Oppgaven dreier seg om Item-Item collaborative filtering, og gjøres som følgende:

1. Regn ut gjennomsnittet av alle brukers ratings for hver film
2. Regn ut likhet mellom ratingene av film nr. 6 og de andre filmene bruker 7 har ratet (dvs. *sim(6, 2) og sim(6, 3)* vha cosinus/pearson-likheten).
3. Bruk sim-verdiene i formell nr.2 til å regne ut estimatet for ratingen. Resultatet blir som følger:

Original Ratings:

	1	2	3	4	5	6	7	8	Avg.
1	1		3			5			3.00
2			5	4			4		4.33
3	2	4		1	2		3		2.40
4		2	4		5			4	3.75
5			4	3	4	2			3.25
6	1		3		3		3.87	2	2.25

Ratings - average ratings for each movie:

	1	2	3	4	5	6	7	8	sim(6, i)
1	-2	0	0	0	0	2	0	0	0.53
2	0	0	0.67	-0.33	0	0	-0.33	0	**0.37**
3	-0.4	1.6	0	-1.4	-0.4	0	0.6	0	**0.05**
4	0	-1.75	0.25	0	1.25	0	0	0.25	0.29
5	0	0	0.75	-0.25	0.75	-1.25	0	0	0.41
6	-1.25	0	0.75	0	0.75	0	0	-0.25	1.00
Svar	3.87471999954169								

r_7_6 = (0.37*4 + 0.05*3)/(0.37+0.05) = **3.87**