Framside

Department of Computer Science

Examination paper for TDT4305 Big Data Architecture

Academic contact during examination: Kjetil Nørvåg and Heri Ramampiaro
Phone: 41440433 and 99027656

Examination date: May 28th

Examination time (from-to): 1500-1900

Permitted examination support material: D: No tools allowed except approved simple calculator. Other information:

Students will find the examination results in Studentweb. Please contact the department if you have questions about your results. The Examinations Office will not be able to answer this.

1 Problem 1 – Hadoop – 10 % (all having same weight)

1. Explain two techniques that together make fault tolerance on data nodes (DataNodes) unnecessary.

1. Replikering. Replikering i HDFS går ut på at datablokkene blir lagret på mer enn en plass. Som oftest er det tre-veis replikering, som vil si at en datablokk vil bli lagret på tre forskjellige noder. Disse kan være på helt forskjellige lokasjoner. Replikering bidrar til feiltoleranse, ettersom dersom en node krasjer, så ligger fortsatt blokkene som var lagret på den noden på andre noder som kan håndtere forespørsler.

2. Sjekksummer. Den fungerer sånn at når en klient skriver til ei fil, så blir det også lagd en sjekksum som lagres som meta-data i den aktuelle datanoda. Når den samme fila skal leses igjen, så sjekker regner datanoda ut sjekksummen for det som ligger i fila og sammenligner med sjekksummen den har lagra i metadataen. Dersom de stemmer overens, så kan fillesingen fortsette, men hvis ikke så har det skjedd noe galt med fila, og man må da lese fila fra en annen replikasjon.

2. Explain *combiner* in MapReduce, and what is the purpose of this.

Hensikten med combiner i MapReduce er å redusere kommunikasjonskostnaden. Dette er aktuelt dersom man har en map-funksjon som produserer mange par med samme nøkkel. Da kan combiner-funksjonen utføre delvis sammenslåing av parene som har samme nøkkel, som oftest ved bruk av samme funksjon som Reduce. Dette kan minke datamengden man er nødt til å sende over nettverket.

3. What is the purpose of using data pipelines when writing blocks in HDFS?

NB! Ligger tekst fra læreren i nynorsk

Hensikten med å bruke data pipelines under skriving av blokker i HDFS er å redusere kommunikasjonskostnaden for klienten, som da kun trenger å sende dataen til ei node i stedenfor til tre (ved tre-veis replikering). Man flytter altså replikeringsbyrden vekk fra klienten. Og i tillegg der ein har fleire rack vil ein redusere kommunikasjon mellom rack (om to replika vert skrive til remote rack vil ein kun ha behov for ein inter-rack kommunikasjon).

2 Problem 2 – Spark – 10 % (all having same weight)

Tuddel is a subscription based streaming service for music. Information about all songs that are streamed is stored in a log, in order to be able to analyze, perform recommendations, and calculate royalties to artists. For each song streamed, a line will be generated in the log, in the following format (comma separated):
TimeStamp,UserName,Artist,SongName

Assume the following example dataset stored in the file streamed.csv:

1,u1,a1,s1
2,u2,a1,s2
3,u1,a1,s2
4,u3,a1,s1
5,u1,a2,s3
6,u2,a1,s2
7,u2,a1,s2
8,u2,a1,s2

This dataset has already been loaded into an RDD named s:

val s = sc.textFile("streamed.csv").map(_.split(","))

For each of the subproblems below, you should show how they can be solved using Spark transformations/actions (Scala, Python or Java).

1. Create an RDD containing number of songs streamed for each artist. Example results:
   (a2,1)
   (a1,7)

```
val new_rdd = s.map(a => (a(2),1)).reduceByKey(_+_)
```


2. Create an RDD that for each line has name of user and number of song he/she has streamed. Example results: u3 1
   u2 4
   u1 3

```
val new_rdd2 = s.map(a => (a(1),1)).reduceByKey(_+_)
```

3. Find number of *distinct* songs that have been played. Example results:
   3

```
val new_rdd3 = s.map(a => (a(3)).distinct().count()
```

Maximum marks: 10

4 Problem 4 – MinHashing – 10 %

Explain the main features of LSH (locality sensitive hashing) for documents, incl.:

1. The purpose of LSH.

Hensikten med LSH er å redusere antall par som må sammenlignes. Uten LSH kan dette ta altfor lang tid. LSH er en metode for å finne kandidatpar som er "like nok" til at de må sammenlignes med en likhetsfunksjon senere.

2. Input and output.

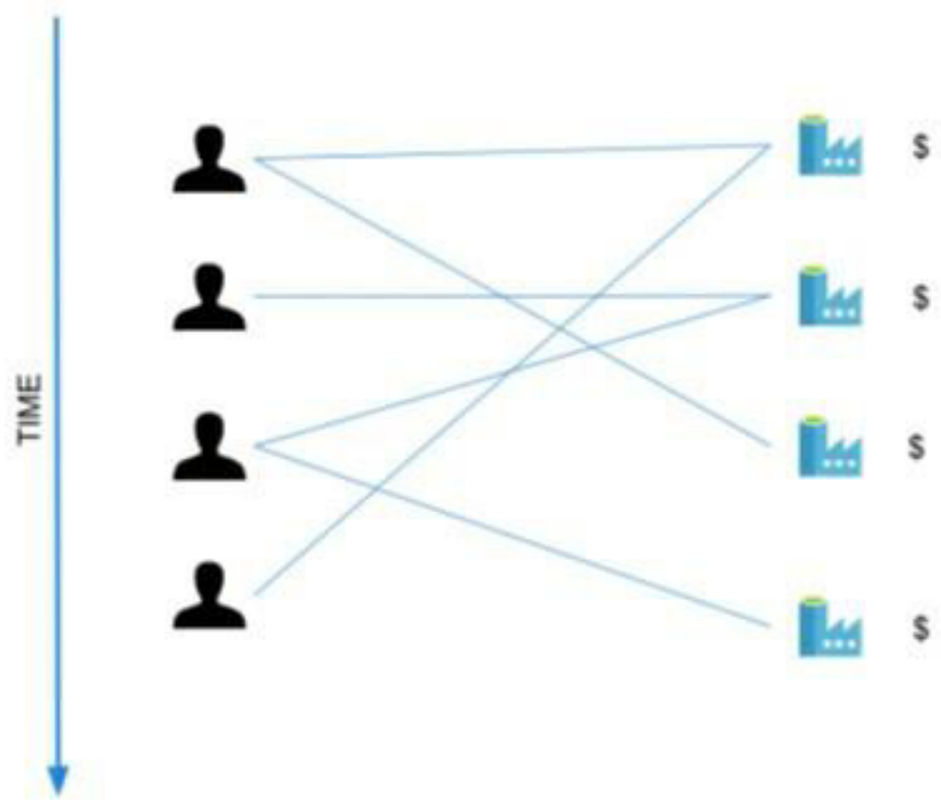
Inputen til LSH er en signaturmatrise, som er generert ved bruk av shingling og minhashing. Output er en liste med kandidatpar som må evalueres av en gitt likhetsfunksjon.

3. Contents of important data structure(s), and algorithm.

Algoritmen er i enkle trekk at man først deler opp signaturmatrisen i b antall bands (bånd), som alle inneholder r rader hver. Deretter går man gjennom hvert dokument for hvert bånd, og hasher disse delene av signaturene til buckets med et visst antall hash-funksjoner. Alle som ender opp i samme bucket etter at man har hashet alle signaturene i hvert band, er kandidatpar. Det at de er kandidatpar, betyr at de må evalueres.

Maximum marks: 10

5 Oppgave 5 – Adwords – 5 %

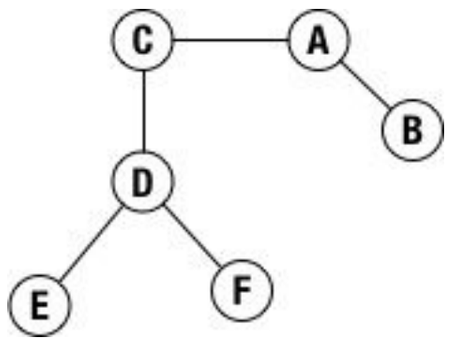


Given the figure above, explain the Adwords problem and a Greedy algorithm for solving this problem. What is the disadvantage of using this algorithm versus an optimal solution? A minor change to the greedy algorithm can improve results. Explain this change.

Fill in your answer here

6 Oppgave 6 – 15%

1. Explain briefly what steps you need to execute to find "communities" in a graph. Use the following figure to support your explanation. (6%)



2. Explain the main purpose of analyzing social graphs. (4%)

3. Explain the main differences between **Storm** and **Spark**. Explain the advantages of the **AsterixDB**'s Feed System has compared to both **Storm** and **Spark**. (5%)

Fill in your answer here

Maximum marks: 15

7 Oppgave 7 – 20%

1. a. What is "Bloom Filter" and what is it used for? Use an example to support your explanation. (4%)

   b. Use Bloom filter to complete the following table. Assume that we use h as a hash-function, and that it is defined as h(x)= y mod 11, where y is extracted from the odd number bits from xor even number bits from x. E.g., h1(39) = 011 = 3,
   h2(39) = 101 = 5, osv. (6%)

Strømelement	Hash-funksjon - h_1	Hash-funksjon - h_2	Filtrere Innhold
			000 0000 0000
85 = **101 0101**			
214 = **1111 1010**			
353 = **01 0110 0011**			

2. Assume that you will find fractions of unique queries from the last month in a stream of search queries. Due to space and time limitation, you cannot check all incoming queries and must use sampling and decide to read only every 10th of the questions (i.e. 10% sampling). Explain why this will not give the correct answer? What would be a more correct approach to sample the stream of search queries? (5%)

3. Imagine you are using social media like Twitter to analyze, e.g., trends. In particular, you are interested in knowing when a product is mentioned by people and how often they are mentioned. You propose using the bucket principle to count how many times this product is mentioned. Explain how the bucket principle on data stream works in this case. (5%)

Fill in your answer here

Maximum marks: 20

8 Oppgave 8 – 15%

1. So-called cold start is a challenge when using collaborative filtering in recommended systems? Explain what is meant by cold start and when this can occur. Further, discuss possible solutions for the cold start problem. (6%)

TDT4305

2. Assume the following user rating table:

USERS

PRODUCTS	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	P	3		3			2

Assume further that you will use "item-item collaborative filtering".

a. Explain what is the difference between item-item collaborative filtering and user-item collaborative filtering. (4%)

b. Explain how you will proceed to compute the predicted / estimated value of P. Make the assumptions you find necessary. (5%)

Fill in your answer here

Maximum marks: 15