Cover Page

Department of Computer Science

Ordinary Exam in TDT4305 Big Data Architecture

Contact during the exam: Heri Ramampiaro
Phone: 99027656

Date for the exam: 27th of May 2019
Time: 09:00 – 13:00

Permitted aids: D – No printed or hand-written support material is allowed. A specific basic calculator is allowed.

Read all questions carefully. Make any necessary assumptions wherever these are necessary.

Problem 1 – Big Data (10%)

1. Along with the term Big Data, we often refer to the three or four V's, and "Velocity" is one of such V's. What does Velocity stand for and why is this important to consider in relation to handling Big Data? Use example to support your explanation.

Answer:
- Velocity stands for speed, i.e., the speed of data creation from a given source such as social media messages and sensors.
- Velocity is important to consider since the data are also normally created in a high speed, the come as open ended and in a large volume, putting high demands on processing efficiency and scalability, in addition to hardware requirements. For example, when computing the average of all streaming tweets or Instagram messages containing a specific topic, we need to consider the fact that the comes in all the time in a speed that might be unbound. This makes it necessary to put some boundary, e.g., in terms of a window size based on time.

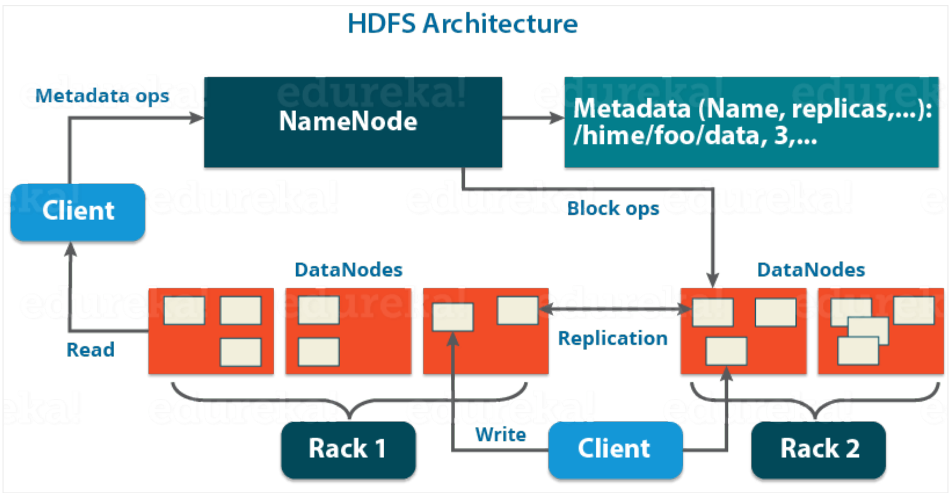
2. Briefly name and explain at least two examples of systems used to handle Big Data.

Answer:
A system to handle big data could be a framework, platform or a software, i.e., Hadoop, Spark, AsterixDB, etc..

Problem 2 – Hadoop (10%)

1. What is Hadoop and explain briefly how the Hadoop Distributed File System (HDFS) is built up.

Answer:
Hadoop Distributed File System (HDFS) is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines. HDFS follows the master–slave data architecture. Each cluster comprises a single Namenode that acts as the master server in order to manage the file system namespace and provide the right access to clients.



2. Explain what the roles of "NameNode" are in HDFS.

Answer:
NameNode is, as can be inferred above, the master node in the HDFS architecture. It maintains and manages the blocks present on the DataNodes (slave nodes). NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients.

3. How is error tolerance taken care of with HDFS?

Answer:
HDFS provides a reliable way to store huge data in a distributed environment as data blocks. To ensure fault tolerance, these blocks are replicated across different data nodes.

Problem 3 – MapReduce og Spark (20%)

1. Explain what the ideas behind MapReduce are (5%).

Answer:

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map function* that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce function* that merges all intermediate values associated with the same intermediate key.

Programs written in this functional style are *automatically parallelized and executed on a large cluster of commodity machines*. The run-time system (e.g. Hadoop) takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication, thus allowing programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

2. In the social media platform, LinkedIn, one often wants to find lists of connections, often called "connections". Such a connection is bi-directional, i.e., if you have a connection with a person, this person will also have a connection with you. Now, suppose, for example, we have the following lists of people and their connection:

   Adrian:Bjørn, Camilla, David
   Bjørn: Adrian, Camilla, David, Ester
   Camilla: Adrian, Bjørn, David, Ester
   David: Adrian, Bjørn, Camilla, Ester
   Ester: Bjørn, Camilla, David.

   Quite often when you try to get in touch with a person on LinkedIn, you want to find out which other persons this person has a common connection with you. This necessitate the ability to look up such information efficiently. Show how MapReduce can be used for this purpose. (10%).

Answer:

Note: *Students have to show they understand the concept. To get a full score, they have to show how map-reduce procedure is concretely done using the LinkedIn connection task. This means that only explaining a solution sketch will not give any full score.*

LinkedIn has a list of *connections*, and as noted connections are a bi-directional – meaning that if I'm your connection, you're mine. When you visit someone's profile, you see a list of connections that you have in common. Assuming this list doesn't change frequently, it would be wasteful to recalculate it every time you visited the profile. This is why MapReduce would be suitable for the calculation of everyone's common connections so that later on it's just a quick lookup, and in our case, this will be a two-names pair, i.e., (name1, name2).

We assume that the connections are stored as `Person: [List of Connections]`, and that they are as follows:

```
Adrian: Bjørn, Camilla, David
Bjørn: Adrian, Camilla, David, Ester
Camilla: Adrian, Bjørn, David, Ester
David: Adrian, Bjørn, Camilla, Ester
Ester: Bjørn, Camilla, David.
```

Mapper:
Each line will be an argument to a mapper. For every connection in the list of connections, the mapper will output a key-value pair. The key will be a connection along with the person (two-names pair). The value will be the list of connections. The key will

be sorted so that the connections are in order, causing all pairs of connections to go to the same reducer. After all the mappers are done running, you'll have a list like this:

For **`map(Adrian: Bjørn Camilla David):`**
```
(Adrian Bjørn): Bjørn Camilla David
(Adrian Camilla): Bjørn Camilla David
(Adrian David): Bjørn Camilla David
```

For **`map(Bjørn: Adrian Camilla David Ester):`** *(Note that Adrian comes before Bjørn in the key)*
```
(Adrian Bjørn): Adrian Camilla David Ester
(Bjørn Camilla): Adrian Camilla David Ester
(Bjørn David): Adrian Camilla David Ester
(Bjørn Ester): Adrian Camilla David Ester
```

For **`map(Camilla: Adrian Bjørn David Ester):`**
```
(Adrian Camilla): Adrian Bjørn David Ester
(Bjørn Camilla): Adrian Bjørn David Ester
(Camilla David): Adrian Bjørn David Ester
(Camilla Ester): Adrian Bjørn David Ester
```

For **`map(David: Adrian Bjørn Camilla Ester):`**
```
(Adrian David): Adrian Bjørn Camilla Ester
(Bjørn David): Adrian Bjørn Camilla Ester
(Camilla David): Adrian Bjørn Camilla Ester
(David Ester): Adrian Bjørn Camilla Ester
```

And finally for **`map(Ester: Bjørn Camilla David):`**
```
(Bjørn Ester): Bjørn Camilla David
(Camilla Ester): Bjørn Camilla David
(David Ester): Bjørn Camilla David
```

Before we send these key-value pairs to the reducers, we group them by their keys and get:
```
(Adrian Bjørn): (Adrian Camilla David Ester) (Bjørn Camilla David)
(Adrian Camilla): (Adrian Bjørn David Ester) (Bjørn Camilla David)
(Adrian David): (Adrian Bjørn Camilla Ester) (Bjørn Camilla David)
(Bjørn Camilla): (Adrian Bjørn David Ester) (Adrian Camilla David
Ester)
(Bjørn David): (Adrian Bjørn Camilla Ester) (Adrian Camilla David
Ester)
(Bjørn Ester): (Adrian Camilla David Ester) (Bjørn Camilla David)
(Camilla David): (Adrian Bjørn Camilla Ester) (Adrian Bjørn David
Ester)
(Camilla Ester): (Adrian Bjørn David Ester) (Bjørn Camilla David)
(David Ester): (Adrian Bjørn Camilla Ester) (Bjørn Camilla David)
```

Reducer:
Each line will be passed as an argument to a reducer. The reduce function will simply intersect the lists of values and output the same key with the result of the intersection. For example, `reduce((Adrian Bjørn): (Adrian Camilla David Ester) (Bjørn Camilla David))` will output `(Adrian Bjørn) : (Camilla David)` and means that connections Adrian and B have C and D as common connections.

The result after reduction is:
```
(Adrian Bjørn): (Camilla David)
```

```
(Adrian Camilla): (Bjørn David)
(Adrian David): (Bjørn Camilla)
(Bjørn Camilla): (A David Ester)
(Bjørn David): (A Camilla Ester)
(Bjørn Ester): (Camilla David)
(Camilla David): (A Bjørn Ester)
(Camilla Ester): (Bjørn David)
(David Ester): (Bjørn Camilla)
```

Now when David visits Bjørn's profile, we can quickly look up (Bjørn David) and see that they have three connections in common, (Adrian Camilla Ester).

3. Explain briefly how you would solve the same problem using Spark. Make any assumptions that you think are necessary. (5%).

Answer/Solution sketch:
Here the student has to show how he/she creates an RDD(Resilient Distributed Dataset) of input strings using the information above. Thereafter they have to explain how the input is used for the map and reduce functions. An answer here could look like a pseudo code.

Problem 4 Systems for data stream (10%)

1. Explain the main differences between Spark and Storm. Discuss especially when Storm is more suitable than Spark and vice versa.

Answer/Solution sketch:
Explanation can be based on the following table:

	Storm	**Spark**
Processing Model	Event-Streaming	Micro-Batching / Batch (Spark Core)
Delivery Guarantees	At most once / At least once	Exactly Once
Latency	Sub-second	Seconds
Language Options	Java, Clojure, Scala, Python, Ruby	Java, Scala, Python

Development	Use other tools for batch	Batching and streaming are very similar

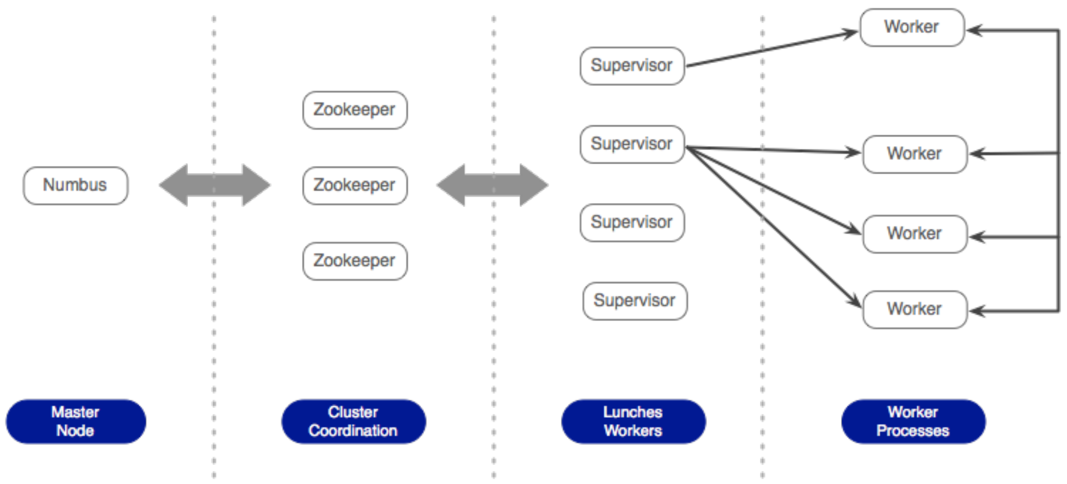
Based on this table, Spark is recommended when we need an iterative (mini) Batch Processing as in most Machine Learning tasks and when the latency requirements is seconds and not less. In contrast, Storm is recommended when we need a real-time processing, with a sub-second latency requirement.

2. Explain Storm's system architecture. Use the figure to support your explanation

Answer/Skecth:

Storm's system architecture consists of several components as shown in the figure below.

- *Nimbus* is a master node of Storm cluster, which is responsible for distributing data among all the worker nodes, assign tasks to worker nodes and monitoring failures.
- *ZooKeeper* is a service used by a cluster (group of nodes) to coordinate between themselves and maintaining shared data with robust synchronization techniques. Since, nimbus is stateless, it depends on ZooKeeper to monitor the working node status. In addition Zookeeper helps the supervisors to interact with nimbus.
- *Supervisor*s are the nodes that follow instructions given by the nimbus. A supervisor has multiple worker processes and it governs worker processes to complete the tasks assigned by the nimbus.
- *Worker* will execute tasks related to a specific topology. A worker process will not run a task by itself, instead it creates *executors* and asks them to perform a particular task.



3. Discuss how AsterixDB can be used to replace Spark and Storm.

Answer:
AsterixDB is a BDMS (Big Data Management System) with a rich feature set that sets it apart from other Big Data platforms. This set of feature includes handling streaming data

through so-called data feed and user-defined function. A data feed here is a flow of data from an external source into persistent (indexed) storage inside a Big Data Management System (BDMS). AsterixDB replaces Spark and Storm in providing the possibility not only to handle inputs of data streams, but also the ability to process the input data in both real-time and through batches, with high performance and scalable UDFs. In addition, AsterixDB provides a tool/system for persisting the input and processed data in a database using the AsterixDB data model.

Problem 5 Social network (15%)

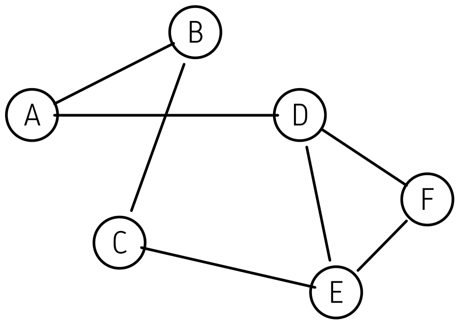


Figure 1 Social network graph

1. What are the main purposes of **community detection** in social networks? (2%)

Answer:
Community detection in social network is key to understanding the structure of complex networks, and ultimately extracting useful information from them. One of its main concern is to detect clusters in a graph making it useful for, e.g., to communality among people connected in the network. This in turns could be used for recommendation or other decision support purposes.

2. Construct an **adjacent matrix** based on Figure 1. (5%)

Answer:
An adjacent matrix *A* of a graph is an *n* x *n* matrix, with which *n* is the number of nodes in the graph, and $A = [a_{ij}]$, $a_{ij} = 1$ if there is an edge between a node *i* and *j*. Based on this the adjacent matrix for our graph in Fig. 1 would be given as in the following table:

	A	B	C	D	E	F
A	0	1	0	1	0	0
B	1	0	1	0	0	0
C	0	1	0	0	1	0
D	1	0	0	0	1	1
E	0	0	1	1	0	1
F	0	0	0	1	1	0

3. What do we mean by **edge betweenness**? Use the Girvan-Newmann method to find the edge betweennesses in the network illustrated in the figure above (Figure 1). Make the assumptions that you think are necessary. (8%)

Answer/Solution sketch:
Edge betweenness is the number of shortest paths passing through the edges. It is used to identify possible communities in a graph. Using the Girvan-Newmann method, we can compute the edge betweennesses in a graph through the following steps:

- Repeat until no edges are left:
    - Calculate betweenness of edges
    - Remove edges with highest betweenness
- Connected components are communities
- This gives a hierarchical decomposition of the network

To get a full score an answer must contain a more detailed description of how to compute the betweenness of edges.

Problem 6 Data Stream 1 (10%)

1. Standing queries are a term often used in relation to querying data streams. Explain briefly what this is about and illustrate with example how this can be used. (5%)

Answer:
Standing queries are queries that are, in principle, asked about the stream at all times. Example: Report each new maximum value ever seen in stream S.

2. Explain how we can use "Bit Counting" to find out the fraction of Instagram users who mention the word "iPhone" in their Instagram messages. Make any assumptions you find necessary. (5%)

Answer:
Bit counting concerns solving the following problem: Given a stream of 0's and 1's, be prepared to answer queries of the form "how many 1's in the most recent k bits?" where $k \leq N$. Translated to our question, we get each time an Instagram message mentions/contains iPhone we set stream bit to "1" if this is the case, "0" otherwise. We will then eventually get a stream of 1s and 0s. To find the fraction Instagram users who mention the word "iPhone", we could then just count the number 1s in the total length of binary stream seen so far.

Problem 7 Data Stream 2 (10%)

1. Explain the principle behind bucket representation of data streams. Use illustration to support your explanation. (5%)

Answer:
A bucket representation of a stream is *a segment of a stream window*, represented by a record consisting of:
- The timestamp of its end [O(log N) bits].
- The number of 1's between its beginning and end, which also constitutes the size of the bucket.

2. Use the "bloom filter" principle to fill in the table below (5%).

Stream elements	Hash-function - ***h1***	Hash- function - ***h2***	Filter content
			00000000000
57 = 11 1001			
214 = 101 1110			

Hint: use *h(x)= y mod 11*, where *y* is computed from the odd and even bits of *x,* respectively.

Answer:
57=1**1** **1**0**01** => h1: 101=5 mod 11 = 5 ⇔ h1=5; h2: 110=6 mod 11 = 6 ⇔ h2=6
(1): 00000110000

214=**101** 1**110** => h1: 1110=14 mod 11 = 3 ⇔ h1=3; h2: 011=3
(2) 00010110000

Problem 8 Recommender Systems (15%)

Musify is a start-up company that specializes in music streaming. Their business idea is to let users stream music freely, given that they give feedback on how they like the music they listen to and that Musify can use this for music recommendation and marketing.

1. Musify has the choice of using content-based recommendation and collaborative filtering. Which of these methods would you prefer? Explain. (5%)

Answer:
A key element here looking at the description is the fact that users give feedbacks on how they like the music they listen to. Based on this it is safe to assume that they could either give explicit feedback or implicit feedback. For simplicity we assume the former, and that this is given through rating the music from 1 to 5. With the availability of user feedback the most sensible solution is to use collaborative filtering (CF). The fact that there is no information about how the music are represented – i.e., which info of the music is available, makes the content-based recommendation a harder choice, though this would be preferred to a lowest degree of cold start issues.

2. Assume that Musify selects "collaborative filtering" and that the table below shows how five of their users (U1 - U5) have "rated" the music (M1 - M5) they have listened to from 0 to 10.

User Is/Music-Id	U1	U2	U3	U4	U5	Cosine(i, 1) (user-user)
M1	7	6	7	4	5	
M2	6	7	0	4	3	
M3	0	3	3	1	1	
M4	1	2	2	3	3	
M5	1	0	1	2	3	

We assume that we will use user-user collaborative filtering. (a) What do we mean by user-user collaborative filtering? (b) What another alternative method do we have? (c) Use cosine equality measure to fill in the table above (10%).

Answer:

(a) User-user collaborative filtering is collaborative filtering with focus on the similarity among users – i.e., their preferences. This means that we recommend a given item/product to a given user based on how similar other users have previously rated this item.

(b) Another alternative to user-user CF would be item-item CF. In contrast to the user-user alternative, item-item focuses on the similarity among the items. This means that we recommend a given item/product to a given user based on how other similar items have previously been rated as compared to this item.

(c)

$$\cos(1,2) = \frac{[7,6,0,1,1]\cdot[6,7,3,2,0]}{\sqrt{7^2+6^2+1^2+1^2}\sqrt{6^2+7^2+3^2+2^2}} = \mathbf{0.93}$$

Using similar computation, we get:
Cos(1, 3)=0,70
Cos(1, 4)=0,90
Cos(1, 5)=0,87

Du skal bruke følgende formel til beregningen din:

$$sim(x,y) = cos(r_x, r_y) = \frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|}$$