TENTAMEN (EXAMINATION)

7

Tentamensdatum/*Examination date:* 1/11-19
(åå-mm-dd/*yy-mm-dd*)

AID-nummer
AID number

Ifylles av student

1	2	8	7		

Completed by student

Ifylles av vakt

1	2	8	7		

Completed by supervisor

Utbildningskod/*Education code:* TDDE31 **Modul/*Module:*** TEN1

Kursnamn/*Course title:* Big Data Analytics

Institution/*Department:* IDA

Jag intygar att varken mobil eller något annat otillåtet hjälpmedel finns tillgängligt under tentamen.
I confirm that no mobile or other non-permitted aids are available during the examination. ☑

Inlämnat: antal lösblad 8 **tentamensformulär** ☐
Enclosed: number of sheets ***exam booklet***

Markera behandlade uppgifter med X/*Mark tasks attempted with an X*

X här/*here*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	X	X	X	X	X	X	—	X	—	X					
Erhållna poäng *Points obtained*	2	2.5	2.5	1.5	2.25	2.5	—	2	—	3					
X här/*here*	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Erhållna poäng *Points obtained*															

Anvisningar/*Instructions*

1. Skriv AID-nummer, datum, kurskod och provkod på varje blad som lämnas in/
   Write AID number, date, course code and exam code on every sheet that is handed in
2. På varje papper får högst en uppgift lösas om inget annat anges/
   Maximum one task per sheet unless otherwise instructed
3. Skriv endast på papprets ena sida om inget annat anges/
   Use only one side of each sheet unless otherwise instructed
4. Numrera de papper som lämnas in/*Number every sheet that is handed in*
5. Använd inte röd penna/*Do not use a red pen/pencil*

Sen inlämning ☐
Late hand in

Klockslag _____
Time

Orsak _____
Reason

Σ Poäng/*Points:* 18.5 Betyg/*Grade:* 3

Examinator/*Examiner:* _____ /17.

LINKÖPINGS UNIVERSITET

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 1287	Datum: *Date:* 1/11 - 19	Blad nummer: *Sheet number:* 1
Utbildningskod: *Education code:* TDDE31	Modul: *Module:* TEN 1	

1

Volume: The amount of data eg. Wikipedia

Variety: Different types of data, for example:
structured: dates, time
unstructured: mix of picture and documents.

Veracity: The trustworthiness of the data. Can you trust the information?

Velocity: The frequency of data that has to be processed, eg. finance Market or Social Media.

2

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 1287	Datum: *Date:* 1/11-19	Blad nummer: *Sheet number:* 2
Utbildningskod: *Education code:* TDDE31	Modul: *Module:* TEN1	

2 a) Vertical scalability is upgrading the existing node. With example more ram, more memory, better cpu:s. Horizontal scalability is adding more nodes into a distributed system.

1/1

b) read scalability means that the system can handle an increased number of read operations without loosing performance.
write scalability means that the system can handle an increased number of write operations without loosing performance.

1/1

c) Data scalability is important within for example a flight system. If the system where not able to handle an increased amount of ~~data~~, it could be devistating.

what kind of data is increasing?

0.5/1

2.5p

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 1287	Datum: *Date:* 1/11-19	Blad nummer: *Sheet number:* 3
Utbildningskod: *Education code:* TDDE31	Modul: *Module:* TEN1	

3 a) the typically implemented queries uses keys to retrieve value. So we would have to loop through the keys until we found the one we where searching for.

b) This would be done by using "Alice" as key add a value with IDs that has Alice as name.
"Alice" → "1, [alice_in_se]"

c) In a document database, the information about each of these persons would be its own document. In a key database only the key has to be unique. But in a document database the name describing the value all has to be unique. within a document.

only one difference

not necessarily

1/1

1/1

0.5/1

2.5p

4 BASE

Basically Available - The system is available basically all the time.

Soft state - The system is always changing even if no input has been made.

Eventually Consistent - This means that if no new inputs are made, then eventually all nodes with that data will have the latest update.

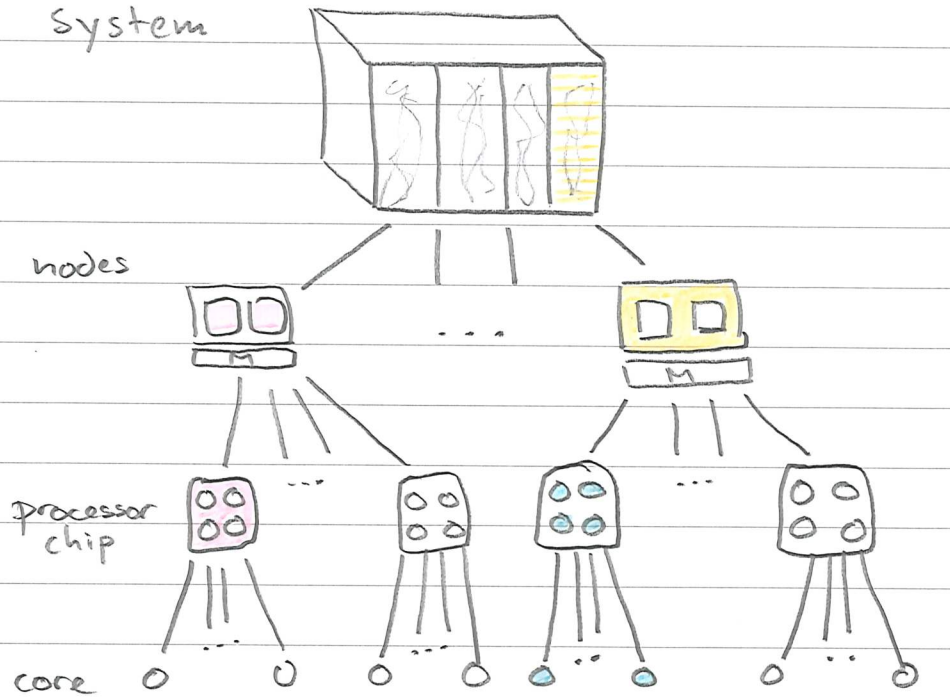
1.5 p

5

a) a distributed file is divided into chunks stored seperatly. A namenode (Master) keeps track on where the parts of a file is saved and the information in each chunk will be handled by datanodes (workers). ✓ Each chunk of a file will be saved at three different places to ensure Fault Tolerance. ✓ Multiple readers makes it good for parallelization ✓

b) /

c)



(✓)

Network?

d) Makes the operations over data more efficient

How?

1,5

0,75

0

2,25

6 a) Who? should not depend on order between rows? of information since the order might differ. where? 0

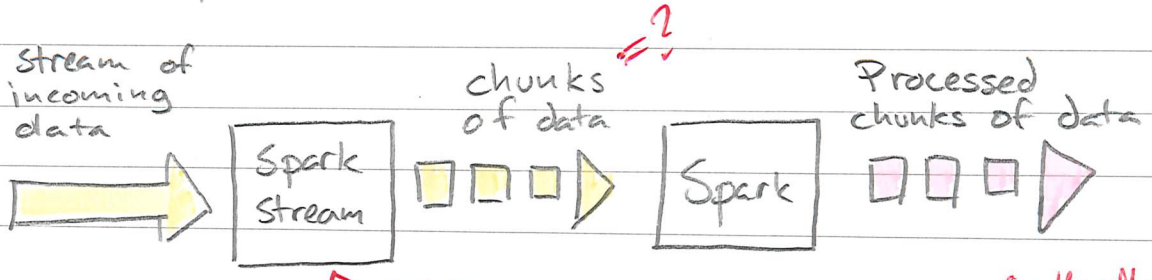
b) first step is the read step, since the collection could be done at a different place this could be network I. ✓
(fourth step is the partitioning. Which) Step 5 sends (shuffle)
the information to the node for reducing phase. which could involve network I/O (✓)
Seventh step is the write which could also involve network O. ✓ 0,75

c) Could be benificial if a lot of duplicate data (more precisely?) could be eliminated before the computition in the reduce step. ✓ 0,75

d)/e) An RDD transformation is an operation that can be done on each RDD (block) seperatly. Example of such an operation is, map() ✗ sort(), ReduceByKey(). 0,5

f) collect picks out the n number of lines from an RDD and leaves the rest of the lines. This collecting a sample. No 0

g) Streaming is a continous stream of data. (?No) Can be used to calculate distance of moving object for example. —
In Spark:



for the effort: 0,5

2,5

8 ① cluster_point = point.map(lambda p: randint(0,1), (1, p))

② total_cluster = cluster_point.reduceByKey(lambda a,b: a[0]+b[0], a[1]+b[1])

average_clust = total_cluster.map(lambda x: x[0], (x[1][1]/x[1][0]))

average_clust.broadcast()

③ for i in range(2)

④ cluster_point = point.map(lambda p: closest_clust(p), (1, p))

⑤ total_cluster = cluster_point.reduceByKey(lambda a,b: a[0]+b[0], a[1]+b[1])

average_clust = total_cluster.map(lambda x: x[0], (x[1][1]/x[1][0]))

④ def closest_clust(p)

if (distance(p, average_clust[0]) ≥ distance(p, average_clust[1]))

return 1

else

return 0

10

assuming:

all_points is all current points

the key is the class

the value is the point

point is the point to classify

k is the number of neigbours we want to check.

Then the code would look like:

```
point_distance = all_points.map(lambda x: 0, (x[0], distance(point, x[1]).
sorted_distance = point_distance.sortBy(lambda x: x[1][1])
closest_points = sorted_distance.take(k)
total_closest = closest_points.reduceByKey(lambda a,b: a[0]+b[0])
new_point_class = total_closest[0][0] /k
```

3

new_point_class does now have the class for which the point should be according to k-nearest neighbours.