TENTAMEN (EXAMINATION)

Tentamensdatum/*Examination date:* 19-08-22
(åå-mm-dd/*yy-mm-dd*)

AID-nummer
AID number

Ifylles av student: 1 8 6 5 | | *Completed by student*

Ifylles av vakt: 1 8 6 5 | | *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 13 **tentamensformulär** ☒
Enclosed: number of sheets ***exam booklet***

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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X här/*here*	X	X	X	X	X	X	X	X	X	X	X	X	X		
Erhållna poäng *Points obtained*	2	1	0	1	0	0	1	1,75	4,25	0,5	5	4	0,5		
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, utb.kod, modul på varje blad som lämnas in/*Write AID number, date, edu.code and module 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:* 21 Betyg/*Grade:* 4

Examinator/*Examiner:* _____

LINKÖPINGS UNIVERSITET

1

Volume - Size of the Data.

Ex: Huge Datacenters by Facebook since they store vast information.

Variety - What kind of Data

Ex: Video, user activity, etc.

Velocity - Speed at which the Data is generated.

Ex: 70 hrs of video uploaded to youtube every minute.

Veracity - "Correctness" of the Data.

Ex: Are you able to trust it when making decisions?

2

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 1865	Datum: *Date:* 19-08-22	Blad nummer: *Sheet number:* 2
Utbildningskod: *Education code:* TDDE31	Modul: *Module:* TEN1	

2 Because we store a lot of information that is being ~~read/accessed~~ by lots of people at the same time, but it is not critical that the data is 100% correct. More important that it can be used by a lot of people.

1P

3

The claim is ~~correct~~. wrong

Since the bottleneck is going to be disk I/O we can not solve this by having a bigger processor/memory. Need to solve by adding nodes that will store information. So we can read from several nodes at once.

may be used to cache more of the read data, which may help to handle more reads

Also, replacing the harddisk of a server by a faster harddisk is another form of scaling up.

OP

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 1865	Datum: *Date:* 19-08-22	Blad nummer: *Sheet number:* 4
Utbildningskod: *Education code:* TDDE31	Modul: *Module:* TEN1	

4

A video streaming site, such as youtube for example. Need to be able to store a lot of information that is going to be read a lot. Solved with large CDNs (Content Distribution Networks).

4

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 1865	Datum: *Date:* 19-08-22	Blad nummer: *Sheet number:* 5
Utbildningskod: *Education code:* TDDE31	Modul: *Module:* TEN1	

5

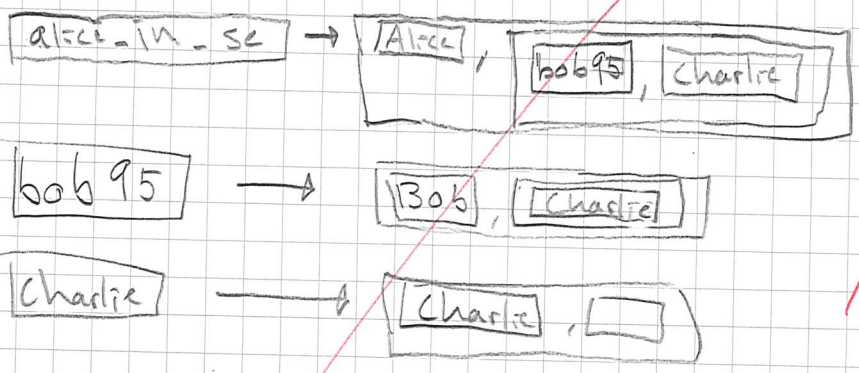
With (key, value) stores we can not index ~~whe~~ the current user likes. Instead this has to be done inefficiently through joining with ~~the~~ other keys.

what are "the other keys"?

joins are not available in key-value stores

OP

6



that's a key-value DB

The form is the something in-between a document-modeled database and a key-value. Compared to a relational database we do not have relations between tables. This enables faster development and more partition tolerance, while the data might be incorrect at some time (even though it will eventually be correct)

OK

rows? row key?
colums?
schema?

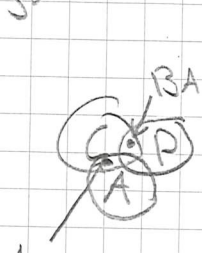
7

a) Consistency - ~~We are always in a consistent state with the database.~~ that's C in ACID

Availability ✓ All requests will receive an answer ~~even~~ though parts of the system is in failure.

~~Partition tolerance - We can partition the data to increase horizontal scalability.~~ wrong

b) The CAP theorem specifies that there is no such thing as a free lunch. We can only achieve 2 of the 3 possible properties for a distributed system. ✓



we have to "sacrifice" one property.

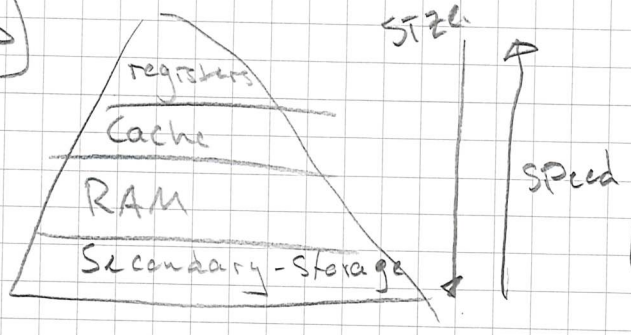
1P

8

a) A distributed file is placed on several nodes. (inodes? No?) In memory where the files "header information" is stored it's pretty similar. However, the pointers (More?) for where data is located is different since the distributed file have the same info stored on several (usually 3) nodes in case one node goes down. ✓

The advantages is: we can store large files that may not fit on a regular disk, (more expensive to create one big disk than several small). different parts of the file can be read in parallel ✓ Also increased parallel opportunities when processing. 1,25

b)



Since creating a single large memory that is fast is expensive and impossible, we use the principles of locality (spatial and local) to achieve this by having several memories in varying size and speed.

The registers are placed inside the CPU-core close to the ALUs.
The cache (or caches with harvard structure) is placed on the CPU chip.
The RAM(s) are seperate components.
Secondary Storage is some kind of Disk (Ex: SSD) or in the case of a distributed file, Disks. + Interconnect? 0,5

c) Since the Disk I/O is the bottleneck we want it to be active all the time to achive higher performance. 0

9

a) Cummutative and associative. [illegible],
Since the order in which the nodes will finish (what?) can't be controlled, we need this to not have an effect on the end result. 1

b) The mapper reads from (?) disk. When remote?
The Shuffle-sort downloads the shards created by the partitioner to each nodes disk
The reducer sends the result to the Output Formatter which writes it back to disk. 1

c) When we need (never mandatory!) to do a local reduce after finishing a map. The reason why this is beneficial is that memory is kept in RAM/cache which increases performance. (+) 0,5
+ reduction of data size!

d) [illegible]. Keep track of old? RDDs to (?) recreate in case of failure(s) 0+
avoiding

e) An operation with an internal global dependece structure. Ex. collect(). ✓ 0,5

f) 

It's a different API based on Spark using DStream. (internally a lot of RDDs)
For in-real-time data input computations ✓

Windowing is the size of how many RDDs being processed at once, used to control how often we start calculations.
↑ Spark

1,25

4,25

10

a) Different programming frameworks can be used at the same time, increased opportunities to manage jobs. ✓ 0,5

b) Because you may want to cancel a certain job if there's a long queue because of it. 0

0,5

11

```
w = uniform(-1, 1)                    <- Initialize w
Training_data.cache()                 <- Assumme we have RDD with
                                         data [tn, xn]
learning_rate = 0.01                  <- static, for simplicity
gradient = 'inf'                      <- Initialize gradient
While gradient > learning_rate:
gradient = Training_data.map(
    (lamba Xi: (1, (-X[0] (1-1/(1+exp(-X[0]*w*X[1])))) X[1])))
    .ReduceByKey(add).collect()

w = w - gradient
```

gradient descent

5

12

```
# Assuming the RDD is globally distributed (broadcasted)
def k_nearest_neighbour(k, point)
    mydata.cache()

    data = mydata.sortBy(lambda x: distance(x[1], Point))
            .take(k)

    Class_0 = data.filter(lambda x: x[0]==0).count
    return  Class_0 > (data.count()-Class_0) ? 0 : 1
```

13 Add a function which takes in a vector/list of points and a k-value. There repeatedly call old function as long as vector/list has not been processed. Store and return vector/list.

Under code/persist? :(