2

TENTAMEN (EXAMINATION)

Tentamensdatum/*Examination date:* 2019-05-29
(åå-mm-dd/*yy-mm-dd*)

	Ifylles av student	Ifylles av vakt
AID-nummer *AID number*	2 2 8 0	2 2 8 0
	Completed by student	*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*

	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					
Erhållna poäng *Points obtained*	2	~~2~~	3	2	1	2,5	~~6,75~~	4,5	4	0					
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:* 21,5 Betyg/*Grade:* 5

Examinator/*Examiner:* _____

LINKÖPINGS UNIVERSITET

LINKÖPINGS UNIVERSITET

AID-nummer: *AID-number:* 2280	Datum: *Date:* 2019-05-29	Blad nummer: *Sheet number:* 1
Utbildningskod: *Education code:* TDDDE31	Modul: *Module:* TEN1	

Q1)

Volume: size of the data e.g. Amazon processes 600TB per day

Variety: type and nature of the data e.g. unstructured, semi-structured

Velocity: speed of generation, processing of data e.g. stock market, social networks

Value: value of the data e.g. insights from Business Analytics are valuable for companies

2

Q2)

a)

Read scalability: a syste can handle increasing numbers of read operations without loosing performance

Write scalability: a system can handle increasing numbers of write operations without loosing performance

✓

b)

A streaming server/application is an example where read scalability is more important than write scalability. Normally, the content is written once but accessed and read by millions of users. ~~Another example would be HDFS which has a single writer and multiple readers.~~

✓

2P

AID-nummer: *AID-number:* 2280	Datum: *Date:* 2019-05-29	Blad nummer: *Sheet number:* 2
Utbildningskod: *Education code:* TDDE31	Modul: *Module:* TEN1	

LINKÖPINGS UNIVERSITET

Q3)

a)

keys	values			
UsMis	[1,000,000	391	350	http://acme.com/r391]
AMee3	[3,700,000	391	350	http://acme.com/r391]
Bee	[1,300,000	121	70	http://acme.com/beerep]
699	[	699	100	http://acme.com/Other]

1/1

b)

Document IDs → Fields consisting of field name and value

UsMis → budget: 1.000.000, final report: 391
AMee3 → budget: 3.700.000, final report: 391
Bee → budget: 1.300.000, final report: 121

} → document

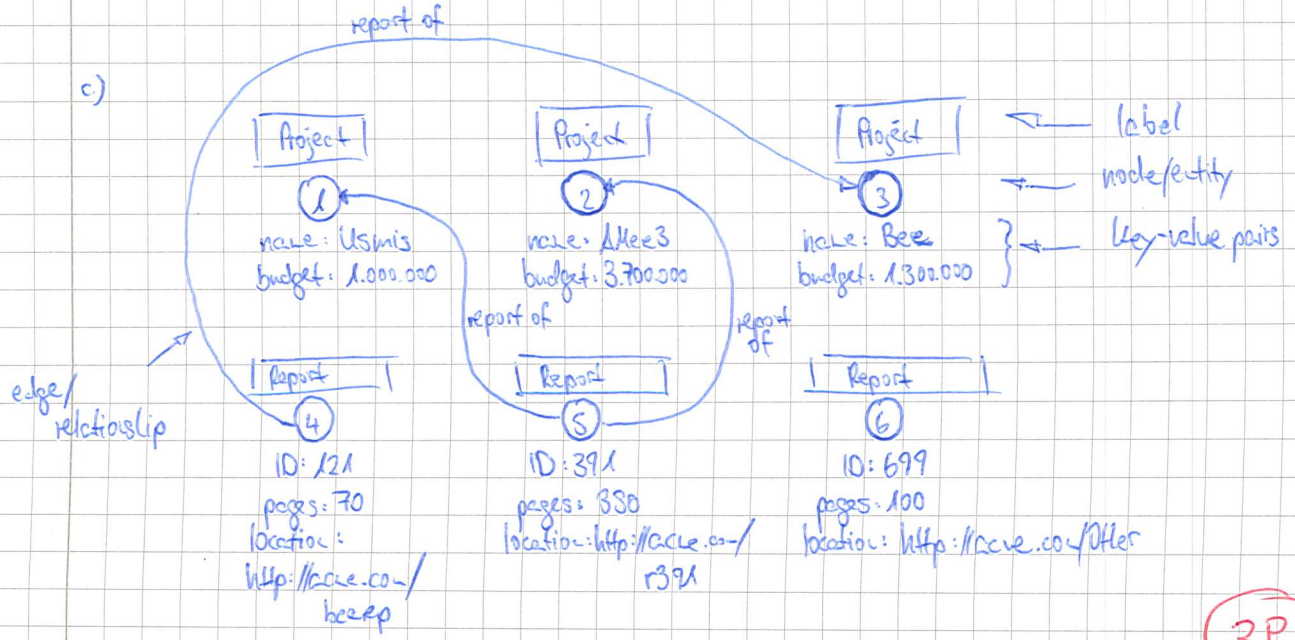
document ~~family~~/collection: Project

121 → pages: 70, location: http://acme.com/beerep
391 → pages: 350, location: http://acme.com/r391
699 → pages: 100, location: http://acme.com/Other

collection: Report

1/1

c)



1/1

3P

Q4)

i) - value-related queries are not possible, since the values are opaque to the syste (no secondary index over values) ✓
- only CRUD operations (create, retrieve, update, delete) regarding the keys are possible
- multiple queries are necessary for retrieving multiple entries

ii) horizontal partitioning/sharding → partitioning based on keys ✓

iii) The limitations of this data model make it very easy to partition the data and distribute the partitions over multiple nodes in a cluster. Horizontal scalability refers to exactly that, the adding of resources by just adding more computers to a distributed syste (scale out). ✓
Key-value pairs are completely independent entities and there is no referencing logic between entities. Therefore, the partitioning based on keys is very efficient and partitions can be distributed without any problems.

2P

Q5.)

Every data item is stored on 3 nodes → N=3

Consistency per CAP requires W=N → W=3 ✓

1P

N: number of nodes that hold a copy

W: number of nodes that need to confirm the successful write operation

(26)

a) A distributed file is split into blocks (eg. 64 MB) and the blocks are distributed over different nodes in a cluster. This allows the parallel processing of the file, since accessing the file can now be done in ~~peer~~ parallel. A traditional file completely sits on one computer and could only be accessed sequetially by multiple computers. + replication for FT 0,5

b) Hybrid clusters are a collection of cluster nodes/computers (DMS) where each node consists of multiple processors sharing the same memory (SMS). Processors within a node and the nodes in a cluster are connected by an interconnection network. The control structure can be characterized by MIMD, multiple instruction and multiple data streams whereas the memory organization can be characterized as a hybrid memory system. The nodes don't have shared memory together so they are refered to as distributed memory system (DMS). They rely on message passing for communication between each other. The nodes themselves are multiprocessors where the processors have shared memory, hence shared memory system (SMS). MIMD means that the whole cluster operates on several data streams and executes different instructions at the same time leading to exploitation of parallelism. 1,5

c) Scheduling tasks (eg. mapper tasks) to nodes that have or are close to the operand data is important because it reduces the communication ~~overhead~~ and the overall load on the interconnection network because the operand data doesn't need to be moved through the cluster. This leads to a better overall performance of the cluster. 0,5

2,5

Q7)

a) Associative: The order of operations is not important as long as the overall sequence of operands is kept the same.

e.g. $(2+2)+4 = 2+(2+4) = 8$

Commutative: The sequence of operands doesn't matter. The result is the same.

e.g. $2+4+2 = 2+2+4 = 8$

These properties are necessary otherwise the Combine and Reduce steps could not be executed in parallel. 1

b)

Record reader: reads blocks from disk and parses them into key-value pairs

Partitioner: partitions the intermediate elements from the mapper/combiner and writes them to local disk

Shuffle-and-Sort: reads the blocks from local disks remotely and copies them to node where reducer is running

Output formatter: receives the final key-value pairs from the reducer and writes them to the global file system 1

c) ~~A combiner is a local reducer running in the mapper task.~~

It is beneficial because it exploits data locality (intermediate elements are still in memory) and reduces the data volume which involved in subsequent steps. This is especially beneficial in situations where the mapper task produces a lot of intermediate elements with the same key. 1

d) An RDD (resilient distributed dataset) is a fault-tolerant collection/container of data elements that can be operated on in parallel. It is ~~distributed and~~ by default lazy, hence it only materializes on demand. The construction of an RDD and its blocks is described by an data flow graph. The blocks are not replicated, in case of a failure the block is simply recalculated based on the data flow graph. RDDs are read-only after construction, partitioned and distributed. 1

e)

Transformations: creates a new dataset (RDD) from an existing one (RDD)

e.g. map, filter ✓

Actions: returns a value to the driver program after running a computation on a dataset (RDD)

e.g. reduce, count ✓ 1

f) Spark has an extension for processing data streams. It uses a high-level abstraction of a continuous data stream called DStream or discretised stream ✓ Essentially, it is a continuous series of RDDs ✓ allowing the operation on RDDs like normally (stream pipelining). + windowing ✓ 0,75

g) Driver: - ~~schedules tasks to workers~~ creates mapper and reducer tasks ✓

- dispatches them to workers → dynamic load balancing ✓

Workers: - execute mapper/reducer tasks from driver ✓

- return the result to the driver (actions) ✓

The client contacts the driver, then the driver (splits the file), creates mapper, reducer tasks and dispatches them to the workers. The workers return their results (actions) or tell the driver where the RDD resides. ✓ 1

Good!

6,75

Q8.)

```
def closestPoint(p, kPoints):
    bestIndex = 0
    currentDist = float("+inf")
    for i in range(len(kPoints)):
        tempDist = distance(p, i)
        if tempDist < currentDist:
            bestIndex = i
    return bestIndex

kPoints = points.takeSample(False, k)
tempDist = 1.0
while tempDist > convergeDist:
    data = points.map(lambda p: (closestPoint(p, kPoints), (p, 1)))
    pointStats = data.reduceByKey(lambda p1, p2: (p1[0]+p2[0], p1[1]+p2[1]))
    newPoints = pointStats.map(lambda p: (p[0], p[1][0] / p[1][0])).collect()
    tempDist = 0
    for i in range(len(kPoints)):
        tempDist += distance(kPoints[i], newPoints["i"])
    for i in range(len(kPoints)):
        kPoints[i] = newPoints["i"]
```

Assumptions:

- collect() returns a dictionary with the elements of the RDD
- convergeDist, k specified by the user
- takeSample(False, k) draws k elements without replacement and returns array
- points is RDD containing the data to be clustered

Q9)

```
def KNN(p, k, data):
    data = data.map(lambda x: (x[0], distance(p, x[1])))
    data = data.sortBy(lambda x: x[1])
    kPoints = data.take(k)
    sum = 0
    for i in range(k):
        sum += kPoints[i][0]
    fraction = sum/k
    if fraction > 0.5:
        result = 1
    else
        result = 0
    return result
classification = KNN(p, k, mydata)
```

Assumptions:

- p, k given by user
- take(n) returns an array consisting of n tuples (key, value)

Q10)

Option 1: Assuming the input p of algorithm KNN is a list of points to be classified one could just iterate through the list of points and do the same map, sortBy, take operations per point to classify → sequential

Option 2: Another option would be to recursively call the function again → One could take the points to classify (assuming they are an RDD as well) and call

```
p.map(lambda x: KNN(p, k, data))
```

This might be better because one could run more tasks in parallel

Code ? 0/5