FINAL

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2018

MEng Honours Degree in Mathematics and Computer Science Part IV
MEng Honours Degrees in Computing Part IV
MSc in Advanced Computing
MSc in Computing Science (Specialist)
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C410H

SCALABLE DISTRIBUTED SYSTEMS DESIGN

Tuesday 20 March 2018, 10:00
Duration: 70 minutes

Answer TWO questions

Paper contains 3 questions
Calculators not required

1 a Briefly explain each of the following concepts, and give an example where each could be applied in the context of scalable distributed systems.

i) Sloppy quorum

ii) Immutability

iii) Energy proportionality

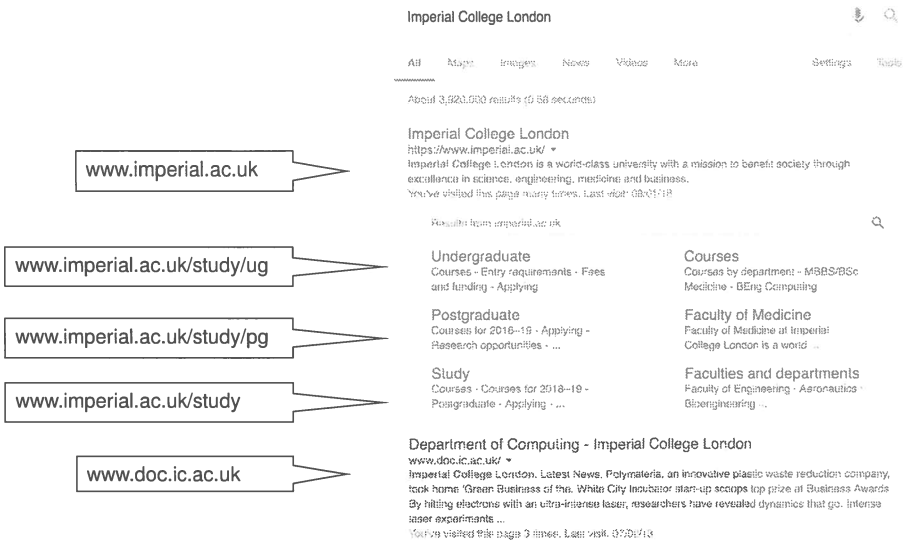
iv) Virtual node

b The **Bigtable** system uses a multi-dimensional sorted map as its data model.

i) Briefly describe, with the help of a figure, the data model used by Bigtable.

ii) Explain why data locality is important for data-intensive systems.

iii) A table maintained by Bigtable stores the data that is returned by a search engine when producing search results. The data, including the corresponding URLs, is shown below:



Write down and justify the data schema for the table stored in Bigtable. You should pay attention to the locality of data accesses.

iv) Explain how you would modify the design of the Bigtable system so that it can further exploit the locality of data accesses in the above search engine example.

The two parts carry, respectively, 20% and 80% of the marks.

2a Briefly explain each of the following concepts, and give an example where each could be applied in the context of scalable distributed systems.

i) Distributed file system

ii) Nearline system

iii) Coordinator

iv) Session

b A **Spark** cluster uses three worker nodes and one master node. The following Spark program is submitted to the cluster:

```
1  val input = spark.textFile("d.csv").persist
2  val cleandata = input.filter(n => n>10)
3  val pointdata = cleandata.map(n => (n, n*n))
4  val result = pointdata.reduce((a,b) => a+b)
```

(Note that you do not need to understand the details of the syntax to answer this question.)

i) State the RDDs and transformations that are used by the above Spark program.

ii) Draw the *lineage graph* for the above Spark program, clearly showing the RDDs and transformations.

iii) When reaching line 3 in the above Spark program, one of the workers fails. Describe all the steps that the Spark cluster will take to recover from the failure.

iv) Describe what a Spark program would look like that exhibits *worst case* behaviour when recovering after the failure of a worker.

The two parts carry, respectively, 20% and 80% of the marks.

3 You work as a software engineer for SMALLDATA, a start-up company that provides solutions for big data processing. Your boss asks you to design a *data-parallel processing system* for a customer who wants to run different analytics jobs over a large amount of data.

The customer has 10 PB of data stored in an *object store*. The object store is deployed on a 1000-node cluster, and the new processing system should run on the same cluster. The object store contains objects with data, and the average size of an object is 400 GB. Each data analytics job will process approximately 10 objects, and generate large intermediate results that are bigger than the original input data.

The object store supports an API to retrieve and store objects based on unique object identifiers, which are URLs. The object identifier is hashed to select a cluster node responsible for storing the object. Objects are also four-way replicated across nodes.

The proposed design of the system should satisfy the following requirements:

(R1) It should perform *data-parallel processing* over the data in order to reduce the time to complete analytics jobs.

(R2) It should *read* the input data directly from the object store, i.e. the customer does not want to copy the data to a different storage system.

(R3) It should *write* intermediate datasets to the object store.

(R4) It should be *fault-tolerant*.

(You should make justified decisions about any aspects that are left unspecified.)

a Explain why using MapReduce or Spark directly on top of the object store does not satisfy all requirements (R1)–(R4).

b Describe the design of a system that satisfies requirements (R1)–(R4), assuming that you **cannot** change the implementation of the object store

Draw a diagram of your system design, clearly labelling all distributed components. For each component, explain its operation and justify its function.

c For each of the requirements, (R1)–(R4), explain how your design achieves it.

d Describe how your answer under (b) above would change if you could make changes to the implementation of the object store.

The four parts carry, respectively, 20% 35%, 20%, and 25% of the marks.