Carnegie Mellon University in Qatar

Distributed Systems

15-440 - Spring 2022

Problem Set 4

Out: March 23, 2022

Due: April 04, 2022

Problem I: Quorum-Based Protocol (20 Points)

Now that you are taking the distributed systems course at CMUQ, you might start getting offers for developing real-world distributed systems. Assume an offer came from *MyBlogPost.com*, a popular website for posting and following blogs, asking you to build a highly-available storage system for them. You accepted the offer and, after much thought, decided to use Gifford's quorum consensus replication as a technique to achieve high availability. You analyzed the user accessibility patterns at *MyBlogPost.com* over the past 12 months and realized that they can be classified into two major categories (say, **CAT1** and **CAT2**). For efficiency reasons, you planned to treat the two categories differently. In addition, to keep your technique simple, you decided to give a weight of 1 to each voting node in the read and write quorums of each category.

5pts

a. Based on your analysis, **CAT1** is read-intensive, with 98% read and 2% write accesses. You are planning to set up a 5-replica system for this category and seeking to optimize for the common case (i.e., make reads fast). What read and write quorum sizes would you select so as to achieve fast reads, while maintaining correctness? Explain your answer.

5pts

b. Based on your analysis, **CAT2** is write-intensive, with 99% write and 1% read accesses. You are planning to set up a 4-replica system and seeking to optimize for the common case (i.e., make writes fast). What read and write quorum sizes would you select so as to achieve fast writes, while maintaining correctness? Explain your answer.

10pts

c. Suppose on each access, a node may fail with a probability, $p = 0.1$. You may also assume that failures are independent. For the read and write quorum values you computed in part B, what is the probability that an access will fail?

Problem II: Programming Models (80 Points)

6pts

a. Explain in detail the differences and relationships between:

(i) A synchronous execution,

(ii) an (asynchronous) execution that uses synchronous communication, and

(iii) a synchronous system.

15pts

b. Suppose you have access to a compute cluster that can run *MapReduce* jobs, as well as a supercomputer that can efficiently run *MPI* jobs. You would like to pick the right tool for the right job. For each of the following, state whether the use case is a better fit for *MapReduce*, *MPI*, or neither. Explain your answer in each case. If *MapReduce* is a better fit, briefly describe what mappers and reducers will do. If *MPI* is a better fit, briefly describe why it wins over *MapReduce*.

i. Compute the average of all pixels in a 1000x1000 image

ii. For each of 10^6 images, compute the average value of all pixels in an image. Each image is of size 1000x1000 pixels.

iii. Process an image of size 10^6 by 10^6 pixels, where the processing of each pixel depends on all of the neighboring pixels, and the processing involves several thousand iterations.

8pts

c. A 50-node Hadoop cluster has a very slow node. Tasks that run on that node always end up being marked as stragglers by Hadoop. A job with 12.5 GB of input data is running on a Hadoop cluster using the default configuration. Assume that each node has 2 Map slots and that Map tasks arrive in waves evenly across all the cluster nodes.

i. How many Map tasks will be launched in total?

ii. How many of those Map tasks will successfully complete and provide inputs to the Reduce tasks?

5pts

d. Which of MapReduce or Pregel is best suited to compute All-Pairs Shortest Paths (APSP) on a weighted, fully-connected Graph? Explain.

18pts

e. The PageRank algorithm is at the heart of Google's search engine. The original purpose for which Google created *MapReduce* was to execute very large matrix-vector multiplications, which are heavily used to rank web pages. In this problem, you shall see that matrix-vector multiplication fits nicely into the *MapReduce* style of computing.

Suppose we have an $n \times n$ matrix M, whose element in row i and column j is denoted as m_{ij}. In essence, M represents the links in the Web, with m_{ij} evaluating to non-zero if there is a link from page j to page i. Alongside M, assume we have a vector v of length n, whose j^{th} element is v_j. Consequently, the matrix-vector product is the vector x of length n, whose i^{th} element x_i is given by:

$$x_i = \sum_{j=1}^{n} m_{ij} \times v_j$$

Clearly, if $n = 100$, there will be no need to use *MapReduce* for performing this matrix-vector product. However, with n in tens or hundreds of billions (as is the case at Google), *MapReduce* can be utilized effectively.

i. Write pseudo-code for the **Map** and **Reduce** functions that can solve the above matrix-vector multiplication, assuming that n is large, but not to an extent that v cannot fit in main memory (i.e., v can be made available to every Map task).

ii. Write pseudo-code for the Map and Reduce functions that can solve the above matrix-vector multiplication, assuming that n is very large to an extent that v

cannot fit in main memory (*hint: think about dividing M and v into different partitions*).

28pts

f. Read the following 2 papers and provide a summary, alongside at least 3 "strengths" and at least 3 "weaknesses" per each paper. Your summary for each paper should not exceed 4 paragraphs.

LA3 Paper
GraphMat Paper