Carnegie Mellon University in Qatar
15440 - Fall 2017

Problem Set 4

Out: October 23, 2017

Due: November 01, 2017

1 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.

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. **(5 Points)**

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. **(5 Points)**

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? **(10 Points)**

2 Programming Models [80 Points]

A. Explain in detail the differences and relationships between: **(6 Points)**

(i) A synchronous execution,

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

(iii) a synchronous system.

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*. **(15 Points)**

(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.

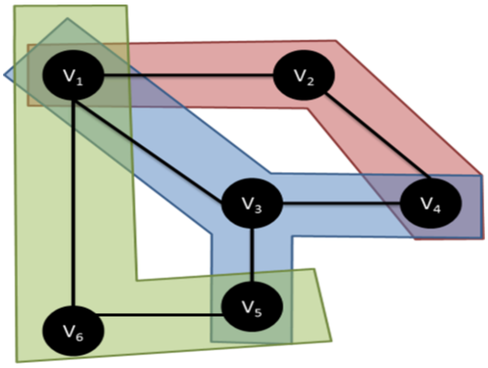
C. A 50-node Hadoop cluster has a very slow node. Tasks that run on that node always end up being marked as stragglers by Hadoop. A job with 12.5 GB of input data is running on a Hadoop cluster using the default configuration. Assume that each node has 2 Map slots and that Map tasks arrive in waves evenly across all the cluster nodes. **(7 Points)**

(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?

D. Which of MapReduce, Pregel, or GraphLab is best suited to compute All-Pairs Shortest Paths (APSP) on a weighted, fully-connected Graph? Explain. **(7 Points)**

E. Given the following graph, assume that vertices are labeled as v_i where i is the ID of a vertex. The red, blue and green areas represent the scopes of the vertices v_2, v_3 and v_6, respectively. Assume an initial vertex schedule $\tau = (v_2, v_3, v_6)$, wherein v_2 was added to the schedule before v_3, and v_3 was added before v_6.



The function $f(v, S_v)$ is defined as $f(v, S_v)$ = the minimum vertex ID among all vertices in S_v. Given that the GraphLab execution engine is run over only one CPU, and can use two types of schedulers: (i) FIFO and (ii) Random, what would the content of τ and the returned value be after the first 3 iterations for each scheduler? Demonstrate the results after every iteration, and discuss which scheduler will guarantee a full coverage of all the vertices after exactly 3 iterations. **(10 Points)**

F. 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.

(a) 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) **(10 Points)**.

(b) 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*) **(5 Points)**.

G. *MapReduce* is a highly successful programming model for implementing large-scale data-intensive applications. However, *MapReduce* is not suitable for many other popular applications, which reuse working sets of data across multiple parallel operations. Read the following paper about a more recent programming model, referred to as *Spark*, which suits these popular applications while retaining the scalability and fault-tolerance of *MapReduce*. Afterwards, compare and contrast *MapReduce* and *Spark* across at least five aspects of your choice (e.g., the execution model). For each chosen aspect, elaborate on how and why *Spark* is similar to or differs from *MapReduce*. **(20 Points)**

Spark Paper