15-440: Distributed Systems

Problem Solving Assignment 4

School of Computer Science

Carnegie Mellon University, Qatar

Fall 2016

Assigned Date: October 31st, 2016

Due Date: November 19th, 2016

I Replication and Consistency

1. Consider the following algorithm that has a complexity of $O(n^2)$ to maintain a causal order across a system with n processes. **(20 Points)**

```
(local variables)
array of int SENT(1...n,1...n)
array of int DELIV(1...n)   // DELIV[k] = # messages sent by k that
                            // are delivered locally
(1)  send event, where Pi wants to send message M to Pj :
(1a) send (M, SENT) to Pj ;
(1b) SENT[i, j] ← SENT[i, j] + 1.

(2)  message arrival, when (M, ST) arrives at Pi from Pj :
(2a) deliver M to Pi when for each process x,
(2b)      DELIV[x] ≥ ST[x, i];
(2c) ∀x, y, SENT[x, y] ← max(SENT[x, y], ST[x, y]);
(2d) DELIV[j] ← DELIV[j]+1.
```

The algorithm works as follows: Each process maintains an $n \times n$ array $SENT$ and an array $DELIV$ of size n. $SENT_i[j,k]$ at process P_i gives the number of messages sent by P_j to P_k, as known to P_i. $DELIV_i[j]$ gives the number of messages delivered from P_j to P_i. In order to avoid violating causal order, a message M that arrives at a process may need to be buffered until all messages that potentially caused M are delivered before M. This is performed in step (2a).

(a) Modify the above causal message ordering algorithm so that processes use only two vectors of size n, rather than the $n \times n$ array.

(b) Is it possible to implement total order using a vector of size n?

(c) Is it possible to implement total order using a vector of size $O(1)$?

(d) Is it possible to implement causal order using a vector of size $O(1)$?

2. A design issue in content replication is whether updates to data items are *pulled* or *pushed* by or to the clients from servers. As discussed in class, the former strategy is commonly known as *pull-based* content replication, while the latter one is denoted as *push-based* content replication. **(16 Points)**

   (a) Discuss one main advantage of pull-based versus push-based replications.

   (b) Discuss one main disadvantage of pull-based versus push-based replications.

   (c) Devise a scenario where neither push-based nor pull-based replication will work well by itself.

   (d) Suggest an algorithm that combines both strategies (or something totally different) so as to effectively tackle your presented scenario.

   (e) When using your algorithm, is it necessary that the clocks of a client and the server are tightly synchronized? Explain.

   (f) Would your algorithm necessarily lead to better performance? Explain.

II Programming Models

1. Explain in detail the differences and relationships between: **(6 Points)**
   (a) A synchronous execution,
   (b) an (asynchronous) execution that uses synchronous communication, and
   (c) a synchronous system.

2. Distributed Operating Systems: **(6 Points)**
   (a) Desktops with tens of cores are already available nowadays. One possible way to harness this power is to parallelize standard desktop applications such as the word processor or the Web browser. Antoher possible way to harness the power is to parallelize the services offered by the operating system (e.g., TCP processing and commonly-used library services like secure http library functions). Which approach appears to be more promising and why?

   (b) Are critical regions on code sections really necessary in an SMP OS to avoid race conditions? Can mutexes on data structures do the job sufficiently? Explain.

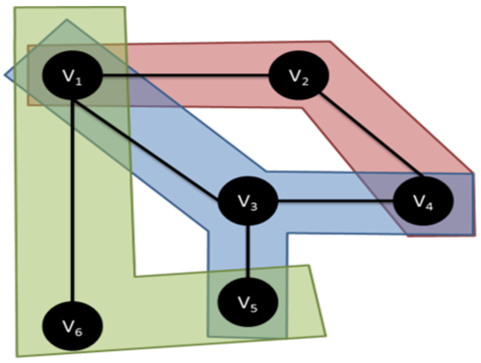
3. A 50-node Hadoop cluster has a very slow node. Tasks that run on that node always end up being marked as stragglers by Hadoop. A job with 12.5 GB of input data is running on a Hadoop cluster using the default configuration. Assume that each node has 2 Map slots and that Map tasks arrive in waves evenly across all the cluster nodes. **(6 Points)**

   (a) How many Map tasks will be launched in total?

   (b) How many of those Map tasks will successfully complete and provide inputs to the Reduce tasks?

4. Which of MapReduce, Pregel, or GraphLab is best suited to compute All-Pairs Shortest Paths (APSP) on a weighted, fully-connected Graph? Explain. **(6 Points)**

5. 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. **(10 Points)**



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.

6. 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) **(BONUS)** 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)**.

7. *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 or differs from MapReduce. **(20 Points)**

   "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing" by Zaharia, et al.