Homework 6 solution

1.

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Each node processes data stored on that node.

Advantages: MapReduce provides automatic parallelization and distribution, fault-tolerance, I/O scheduling, status and monitoring.

Disadvantages: Data must be stored before computing, a large number of network traffic in mapreduce job.

3.

Data distribution:

Distributed file systems stores files on a single storage node.

Parallel file systems distribute data across multiple storage nodes.

Fault-tolerance:

DFS take on fault –tolerance responsibilities

PFS run on enterprise shared storage

Workloads:

DFS are geared for loosely coupled, distributed applications

PFS target HPC applications, which tend to perform highly coordinated IO accesses, and have massive bandwidth requirements.

Symmetry:

DFS run on architectures where the storage is co-located with the application PFS run on on architectures where storage is physically separate

4.

HDFS splits huge files into small chunks known as blocks. These are the smallest unit of data in a filesystem. We (client and admin) do not have any control over the block like block location. Namenode decides all such things. The default size of the HDFS block is 128MB which you can configure as per your requirement. All blocks of the file are the same size except the last block, which can be either the same size or smaller. The files are split into 128 MB blocks and then stored into the Hadoop file system. The Hadoop application is responsible for distributing the data block across multiple nodes.

5.

1.HDFS achieves fault tolerance mechanism by replication process. In HDFS whenever a file is stored by the user, then firstly that file is divided into blocks and then these blocks of data are distributed across different machines present in HDFS cluster. After this, replica of each block is created on other machines present in the cluster. By default, HDFS creates 3 copies of a file on other machines present in the cluster. So due some reason if any machine on the HDFS goes down or fails, then also user can easily access that data from other machines in the cluster in

which replica of file is present. Hence HDFS provides faster file read and write mechanism, due to its unique feature of distributed storage.
2.NameNode uses heartbeats to detect DataNode failure.

6.

- (a) The algorithm does p-1 steps and in each step it sends and receives a message of size m. Therefore the communication time is $(t_s + t_w \text{m})(\text{p-1})$
- (b) Takes log(p) steps for a p-processor hypercube. Therefore the communication time is $t_s \log(\text{p}) + t_w \text{m}(\text{p-1})$
- (a) If the per-word-transfer time is $\text{k}t_w$, and $t_s = 100 * t_w$, so for ring algorithm, the communication time is $(100t_w + \text{k}t_w\text{m})(\text{p-1})$, for hypercube algorithm, the communication time is $100t_w\log(\text{p}) + \text{k}t_w\text{m}(\text{p-1})$.
  The term associated with t_w in the expressions for the communication time of all-to-all broadcast is same for all architectures. This term also serves as a lower bound for the communication time of all-to-all broadcast for parallel computers on which a node can communicate on only one of its ports at a time. This is because each node receives at least m(p - 1) words of data, regardless of the architecture. Thus, for large messages, a highly connected network like a hypercube is no better than a simple ring in performing all-to-all broadcast or all-to-all reduction. So all of the algorithms presented above are asymptotically optimal in message size.
- (b) All of the algorithms presented above are asymptotically optimal in message size. The reason is same as the previous question.

7.

- $\text{t(m)} = \text{t(m-1)} + m^2$

8.

- Log(P) is a model of a distributed-memory multiprocessor in which processors communicate by point-to-point messages. It can approximate memory communication in parallel systems with a fixed overhead parameter (o), the reciprocal of the bandwidth between application and network buffers.
- Because we need a model to capture the important bottlenecks of parallel computers with a small number of parameters and reflect the major practical issues in parallel algorithm design.
- No. Using a single hardware bandwidth parameter to model memory communication performance in a hierarchy is not sufficient since the effective latency overlap of a hardware implementation is application and system dependent.

9.

- Memory-Log(P) is a simple and useful model of point-to-point memory communication to predict and analyze the latency of memory copy, pack and unpack.

- Using a single hardware bandwidth parameter to model memory communication performance in a hierarchy is not sufficient since the effective latency overlap of a hardware implementation is application and system dependent. Models of communication that incorporate these characteristics are warranted when memory communication has significant impact, however resulting models must remain simple despite the complexity of current memory hierarchies.

- No, C-AMAT model integrates the joint impact of locality concurrency, and overlapping for optimization, C-AMAT is a more comprehensive model. Memory-Log(P) is focus on the impact of memory communication.

Bonus Questions

1.

- From the Amdahl's law, the speedup is $S_p = \frac{1}{a+\frac{1-a}{p}}$, here a = $\frac{W_S}{W}$
  If p is infinite, $S_p = \frac{1}{a} = \frac{W}{W_S}$, so W/W_S is an upper bound on its speedup, no matter how many processing elements are used.

2.

- Because remote job has low priority and Local job arriving and service time based on extensive monitoring and observation.

- Leutenegger and Sun determined the capacity of non-dedicated homogeneous computing environments. They considered a discrete model where the machine owners use their machines with a fixed probability and fixed job length.
  Kleinrock and Korfhage used Brownian motion to approximate the parallel task completion time in a non-dedicated system.

- Kleinrock and Korfhage's model assumes parallel tasks arrive equally during each state of the local sequential processing. They derived analytical expressions of the approximated mean and standard deviation of parallel completion time.