Birla Institute of Technology & Science, Pilani
Work Integrated Learning Programmes Division
Second Semester 2022-2023

Mid-Semester Test – Model Solution
(EC-2 Make-up)

Course No.	: AIML CLZG516
Course Title	: ML System Optimization
Nature of Exam	: Open Book
Weightage	: 25%
Duration	: 2 Hours
Date of Exam	: 23/07/2023 (AN)

No. of Pages = 2
No. of Questions = 6

Note to Students:
1. Please follow all the *Instructions to Candidates* given on the cover page of the answer book.
2. All parts of a question should be answered consecutively. Each answer should start from a fresh page.
3. Assumptions made if any, should be stated clearly at the beginning of your answer.

1. Given a large collection C, of N documents, a clustering algorithm works as follows:
   - Step 1: extract keywords from each document, compute distances between pairs of documents, and use the distance metric to decide whether a pair goes in the same cluster or into two different clusters;
   - Step 2: once clusters are formed, compute distances between the centers of the clusters to decide whether or not the two clusters should merge into a larger cluster
   - Step 3: repeat step 2 until either all documents are in one big cluster or distances between cluster centers converge

   Assume you have the following functions:
   - **Vector extract(Doc d)** // extract keywords from d and return a vector
   - **float similar(Vector v1, Vector v2**) // compute similarity between v1 and v2
   - **Doc getCenter(DocSet D)** // return the center of a cluster

Show how this clustering algorithm can be implemented using *map-reduce*.
Calculate the speedup on a distributed system with N nodes.
Analyse the impact of communication cost on the speedup.

[Expected Time: 45 minutes **9 Marks]**

Solution:

map-reduce implementation:

1.1 vectors = (map extract C)

1.2 compute distances : (map similar (vectors X vectors))
// where X denotes a cross-product operator on sets

1.3 if (distance(doc1, doc2) < threshold) then {

1.4 add both documents go in the same cluster }

// clusters formed at this stage

2.1 centers = (map getCenter Clusters)

2.2 compute distances : (map similar (centers X centers))

2.3 if (distance(center1, center2) < threshold) then {

merge the two clusters }

3 repeat step 2 until (*distances converge* OR *Clusters is a singleton*)

Speedup and Communication Cost

- Map will provide speedup linear in N, for steps 1 and 2 but the repeat cannot be parallelized.
-
- Also, the two cross-product steps 1.2 and 2.2 will require communication:

1 All |C| documents sent to all nodes in 1.2

2 All clusters (centers) sent to all nodes in 2.2

3 Map does not require any communication by itself.

- The total messaging cost involved is proportional to **O(|C|+k)** which is **O(|C|),** Where **k** is the number of clusters (this is data-dependent but **k << |C|**)

2. Compare and contrast the speedup of a task-parallel algorithm for **building an ensemble model** for a regression task *on a shared memory system vs. on a distributed system*.

[Expected Time: 15 minutes 3 Marks]

Solution:

- A task-parallel or model-parallel algorithm for building an ensemble model will run different algorithms A1, A2, …, Ak on different processors. The algorithms will all use the same training dataset.
- In a shared memory system, the training dataset will be in global (shared memory) and different processors P1, P2, …, Pk will all have access to the training dataset.
    - As the tasks Aj are different, they may result in uneven workload. The speedup will be constrained by the longest running of these algorithms.
        - Speedup(k) = k, ideally if all algorithms take roughly the same time but if the longest among Aj takes c times the average time, the speedup will be k/c

- There may also be memory – contention among the processors as they all may access shared memory (although the only shared data is accessed in read-only mode).
- Thus the effective speedup will be (k/c)*M_s where M_s is a slow-down factor due to memory contention.

- In a distributed memory system, the training dataset will be replicated in different nodes N1, N2, …, Nk.
  - The replication will result in some one-off communication overhead, which we will ignore in speedup calculation.
  - The speedup will be k/c (see above), where the longest job has a running time c times the average.

3. Compare and contrast **deploying an ensemble classifier** on a multi-core system vs. on a commodity cluster with respect to *throughput*, *response time*, *scalability*, *extensibility* (of the model), and *cost*.

[Expected Time: 25 minutes **5 Marks]**

Solution: An ensemble classifier will in turn use multiple classifier models M1, M2, … Mk each of which is deployed on a different processor; on an incoming sample (or test) data. The classes predicted by these different classifiers will be aggregated (by, for example, a majority vote).

[In a multi-core system, the different models Mj will be run each in a separate thread (i.e., on a separate core).

In a commodity cluster, the different models Mj will be run each on a separate node. The communication required for aggregation is a single message from each of the k nodes.]

- Throughput would multiply by k on both systems.
- Response time will be improved on the cluster as nodes respond completely independently (and the majority decision may be done on any node which can then respond to the request) whereas in a multi-core system the single communication channel is shared by all processors.
- Scalability – where the number of incoming requests increase – but we maintain throughput and response time – is better with the cluster (by adding extra nodes, which replicate the ensemble).
- Extensibility – where the number of models in the ensemble increases – is also handled better on a cluster (by adding extra nodes).
- Typically, ensembles have a few models i.e., k is small. So the cost is not affected by k. But if incoming requests are large in number, clusters are more cost-effective. And if one or more models are large, then again clusters are cost effective.

4. Consider the training phase of ***bagging*** **(**or **bootstrap aggregation)** parallelized on a multi-core system with **m *bootstrap samples***:

   Will it affect the system performance parameters (**not** *machine learning performance attributes* like accuracy) if **(i)** a different bootstrap (or selection) technique is used for each sample **or (ii)** a different learning algorithm is used for each sample **or both (i) and (ii) ?**

   Explain your answer.

[Expected Time: 10 minutes **3 Marks]**

Solution:

(i) Using different bootstrap techniques will not significantly affect system performance

(ii) Using different learning algorithms may result in different times of completion (on different processors) resulting in reduced speedup.

(iii) The effect is the same as in (ii).

5. Argue that the *parameter server* model would be easier to implement and will give better performance on a shared memory system as opposed to its implementation in a distributed system.

[Expected Time: 5 minutes **1 Marks]**

Solution:

Parameter updates will happen on shared memory and hence will be faster (as there is no communication delay) and will result in better performance.

Training data distribution is not necessary as it is stored in shared memory. Each process can access different regions of data and process them in parallel. This makes it slightly easier to implement.

6. Consider the following equation for gradient descent:

$$w = w - \eta * g(L, D, w)$$

(*where g is the gradient function, L the loss function, and D, the dataset and* η *denotes the learning rate)*

and the corresponding pseudo-code for **stochastic gradient descent**:

```
for i = 1 to num_iter {
        shuffle ( data );
    for example in data {
        grad = eval_gradient ( loss_function , example , w );
        w = w - learning_rate * grad;
    }
}
```

Can the inner loop be implemented as a software pipeline? Why or why not?

If the inner loop is implemented as a software pipeline, explain the cost overhead in running this implementation on a distributed system as opposed to a shared memory system.

[Expected Time: 20 minutes 4 Marks]

Solution:

- Each iteration of the inner loop depends on the update from the previous iteration.
- So a 2-stage pipeline can be conceived where one sage is performing weights-update while the previous stage is computing gradients. But this is a mismatch in workload.
- So gradient evaluation stage can be performed in parallel (like mini-batch SGD) but the update with the averaged gradient can be performed. This is strictly not software pipelining alone.
- Furthermore if it were implemented in a distributed system there will be communication overhead in distributing data and for performing updates.
-