Karl Roth
NPRE
1828

Section 1:

1. The efficiency of a parallel program is evaluated by how effectively you're making use of multiple processors. 100% efficiency would imply mean that adding 'p' processors would result in a factor of p speedup. Speedup is defined as the execution time on a single core over the execution time on p cores. Efficiency is the speed up divided by p. The two main challenges associated with parallelism are: load imbalance and not enough parallelism (or data to small for parallelism). If load imbalance is present, then you are putting an uneven amount of work on one processor over the others. This leads to inefficiency because if you aren't spreading the work out evenly. The other processors will finish long before one of them (when they could be contributing to the work). If there isn't enough parallelism, it means that you are processing a task on one work station when the work could be divided. And if the data is too small for parallelization you are wasting time and computational power trying to allocate between machines and organize the data, when it could be processed reasonably quickly on one machine.
2. Message passing is required when two processors do not share a common memory (ie. A distributed memory scenario). When the computing nodes use their own local memory during computation, a message must be passed between the two processors to transfer relevant data between processors This transfer requires specifying the sender, receiver, and the mode of transfer. This has the advantage of being flexible and offering a higher performance, but can be more complex to code. One example of his is the Message Passing Interface, which passes the data from memory through a node interconnect.
3. Synchronization is important to ensure that concurrent threads do not execute simultaneously. If the processes aren't properly synchronized a race condition may occur where the value of variables may change unpredictably depending on which process completes first. For example, if three processes are executing simultaneously and they share a common resource, synchronization should be used to avoid conflicts in accessing this shared resource. In other words, the resource should only be assigned to one process at a time.
4. 4 general categories: Single Instruction Single Data (SISD), Single Instruction Multiple Data (SIMD), Multiple Instruction Single Data (MISD), and Multiple Instruction Multiple Data (MIMD). An example of SISD is a pipelined processor that takes in one stream of data and processes it on a single core. Most modern CPU designs include SIMD instructions. This means that there are multiple processing elements performing the same operation at the same time. MISD occurs when different operations are performed on the same data. There aren't many examples of MISD, but the Space Shuttle flight control computers used this method. Finally, MIMD machines have multiple processors independently perform different instructions. NVIDIA graphics card is one example that fits the MIMD model.

Section 2:

1. Map Reduce is a parallel programming paradigm that is well suited for certain tasks. There are three main stages map, shuffle, and reduce. Performs filtering and sorting, the shuffle appropriately distributes the mapped items, and the reduce function performs a summary operation. The most basic example of this is the word count example. A large text file is broken down into individual words and each word is assigned a value of 1 during the mapping phase. For example: (word, 1). These words will be grouped by word and distributed across machines, and each machine will perform a reduce function which will combine similar words. For example: (word, 20). The end result will be the word and the total number of instances of that word in the entire text file.
2. The HDFS system consists of NameNodes (the master) which manage the filesystem namespace and the DataNodes (workers) that store and retrieve the chunks of data. Additionally, the JobTracker is responsible for scheduling the job's component tasks on the TaskTrackers, monitoring them and re-executing the failed tasks. Each DataNode has a TaskTracker that executes the tasks as directed by this JobTracker.
3. The following command is used to combine and output all partitions in hdfs:
   `hdfs dfs –getmerge file_in_HDFS path_in_local_directory`
4. The following command will transfer a file called "exam2.txt" from the home directory on ROGER filesystem to the home directory on HDFS file system:
   `hdfs dfs –copyFromLocal ~/exam2.txt exam2.txt`
5. The mapper file was edited from the one used in lab10. The delimiter was changed from a comma to 'x01', the date was filtered by all tweets that include "Feb", retweets were filtered out, and the latitude and longitude were split. A lat/lon key was generated and the output was fed into a reducer that is very similar to the word count. It created a list of lat lon keys and summed together the different values for each key.

Section 3:

1. Includes a wide variety of operators like join, sort, filter, etc. Easy to program compared to Hadoop, similar to SQL. Pig helps focus on the optimization, so that the programmer only needs to focus on the workflow. Can write user defined functions in python to perform complex tasks like averaging, and it handles a wide variety of data. Pig is generally used as an abstraction over MapReduce with Hadoop.
2. There are two types, scalar and complex. Among the scalar types is int, long, float, double, chararray, bytearray, and Boolean. Some complex types include, Map (a collection of keyvalue pairs), Tuple (ordered set of fields), and Bag (unordered collection of tuples).
3. FILTER is used to select rows that you want or filter out rows that you do not want based off of some criteria. STORE sends the output to a folder in hdfs. Examples:
   ```
   step1 = FILTER step0 BY $4 MATCHES 'cpm';
   STORE results into heatmap_output;
   ```
4. The script heatmap.pig was created and run. It loads the data into pig, filters out null data points, filters out entries that don't reside in the bounding box, creates a lat/lon key, and averages the fare_amount field over each lat/lon key. The average occurs by passing the data through average.py. The result is saved as heatmap_output in hdfs.

Section 4:

1. The code spark_Query.py was run in pyspark on the ROGER super computer. The file had two UDF's makeIndex and write_to_grid. The make idex function filtered out bad data entries and only allowed latitude and longitudes from within the "bounding box". It then created a lat/lon key for each entry and assigned it a value of 1. This function was then called during the flat map, and then a simple reduceByKey(a+b) was used to sum together all complaints for a given lat/lon key. The result was then passed through the write_to_grid function where the lat/lon key text file was converted into and ASCII file.
2. To ask Apache Spark to retain a particular dataset in memory, perform the following command on the dataset called rdd:
   rdd.cahce()
3. A resilient Distributed Dataset is the primary data abstraction in apache spark. It is an immutable distributed collection of objects. Each dataset is divided into logical partions, that can be distributed across different nodes of the cluster. RDD's can include Python, Java, or even Scala objects, including user-defined classes. RDD's are fault-tolerant collection of elements that can be operated in parallel.

Section 5:

1. The ACID acronym stands for: Atomicity (all or nothing), Consistency (moves from one valid state to another), Isolation (transaction execution can be serialized), and Durability (tolerant against failure).
2. The CAP theorem for NoSQL states that given many nodes and the nodes contain replicas of partitions it is impossible for a distributed data store to simultaneously provide more than two of the following:
   Consistency – all replicas contain the same version of the data, and the client has the same view of the data for any given node.
   Availability – The system remains operational on failing nodes, all clients can read and write.
   Partition Tolerance – the system has multiple entry points, remains operational under communication malfunction, and works well across physical partions.