Cluster and Cloud Computing Assignment 1 - Report

Yasser Aldwyan (606486)

The algorithm:

The implementation of the parallelized search application can be perceived by the following algorithm:

Given Number of nodes (n), processor per node (ppn), so number of processes (p = n * ppn)

1- Split up the data into p parts. Partial data = size of data / p
2- Each process:
    a. Calculate the start position and the end position for the text to be read from the text file.
    b. Start from start position searching for a given term counting the number of matches.
    c. Continue searching until the end position is reached.
    d. Return the partial result.
3- Calculate the final result.

The invocation of the application:

Basically, the search application consists of two files: a source file, StringSearchMT.java and a PBS script, stringSearch.pbs.

Running the application can be done by typing the basic PBS command, qsub and passing the resources that should be allocated, the file including the path and the term that the application will search for and count the number of times it occurs if found. For instance,

```
qsub stringSearch.pbs -l nodes=2:ppn=4 -v term=water, path=/home/yaldwyan/data/CCCdata-small.txt
```

Error Handling:

Regarding hyphenated words, the search application does not support the hyphenated words. Some suggestions are discussed here.

If a given term is a hyphenated word, the search system should consider this and be tolerant with a hyphen during the search process in order to be accurate. In addition, the system should be able to add this given hyphenated word to a hyphenated word list, a dictionary; it is kind of machine learning. The hyphenated word list may have the more likely to be hyphenated words.

In case a hyphen found at the end of a line, the system should be able to merge the word before the hyphen with the first word in the next line, and then check if there is a match. If there is a mismatch the merged word should not be added to the hyphenated word list.

In terms of issues with the accuracy of the timing measurements, variations in performance of the system running multiple times with the same term to search for and the same resources show that there might be subtle factors affect the accuracy of measuring the performance. Graph in Figure 1 shows two examples of these variations in performance of the application where the term is water in the first example and the county term was the other one. It is noticeable that in the first run for the term water, the execution time is 80968 milliseconds. In the second run, however, there is a dramatic drop in the execution time. Then, the execution time rises again in the third run. In the second example, the difference in performance between the second run and the third run is approximately 40000 milliseconds.

So what are the factors that make these variations happen? One explanation may be the location of the data. If most data is still in the cache or at least in the main memory, then the performance will increase since there is no need to fetch a substantial data from the hard disks during the job.

Another explanation may be communication failures while fetching the data to be processes. As a result, the latency will increase. The other explanation may be the interruption of the running job's node in order to know whether the job is finished.

It should be much accurate running the search application many times, and then calculate the average.

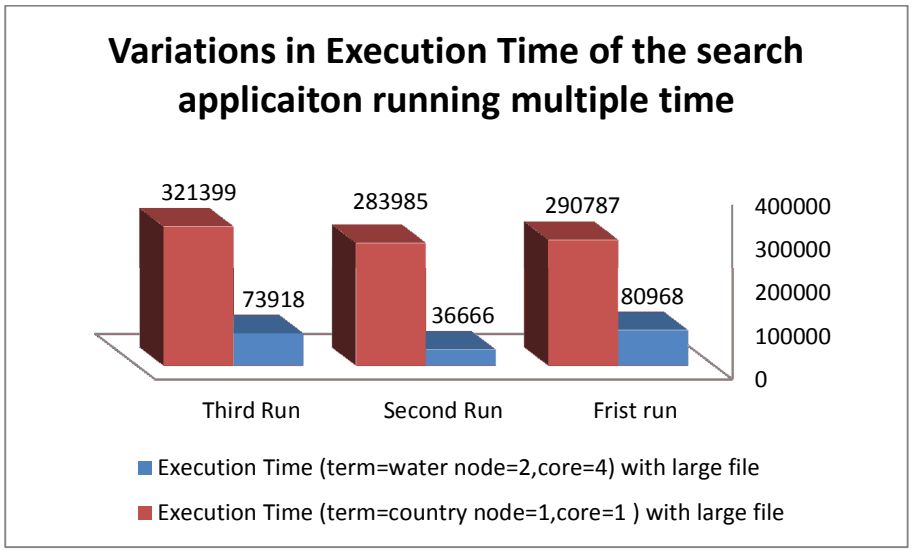

Figure 1 Variations in Execution Time of the search applicaiton running multiple time

The performance of the application on various numbers of nodes and cores with different text files' sizes:

1. **The execution time of the application with small size of text files.**

The chart in Figure 2 presents information about the performance of the search application running on different number of nodes and cores in cluster machines and PC with relatively small size of file (~100 MB). The given term is country and the number of matches is 5767 matches. From this chart some observation can be made. Starting from the single node with a single core in the cluster, the cost in terms of time is very high even though the size of the file is relatively small. Moving to the single node with 8 cores as well as the 2 nodes with 4 cores, a dramatic improvement in performance of the system has been made. In addition, the performance of the application running on PC (intel CORE i7) on either one core or 8 cores are obviously better comparing with the single cluster machines.

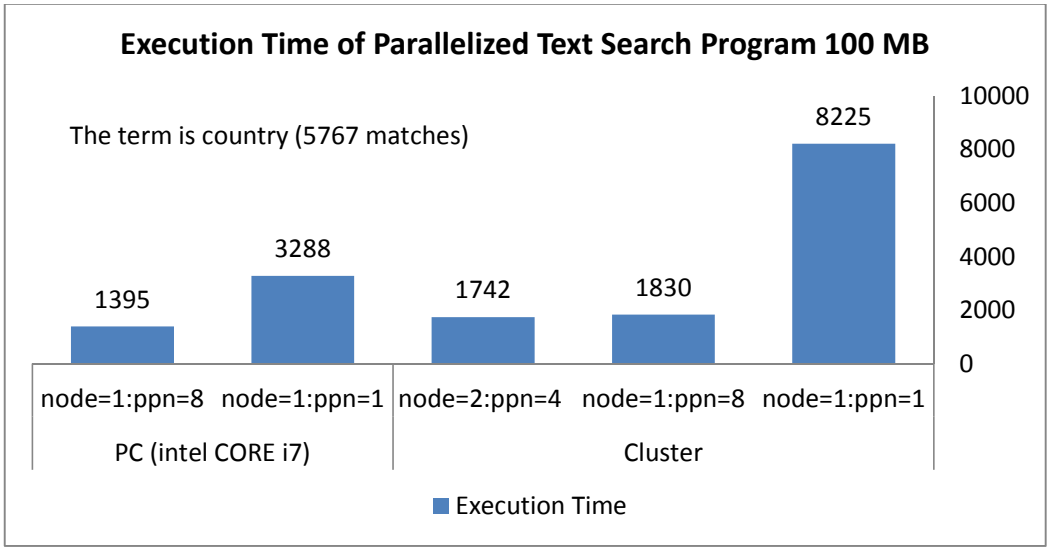

Figure 2 Execution time of search program with small size of text

2. **The performance of the application with medium size of text files.**

The graph in Figure 3 shows data about the execution times of the search application with medium size of file (~1 GB). The number of matches of the given term, country, is 52487 matches.

The performance of the system on the single node with one core in the cluster is significantly low. Its performance on multiple cores in either the single node or two nodes in the cluster has been enhanced. In the PC, the execution time of the application in the single core is still lower comparing with the cluster one. However, in case of multiple cores, the cluster machines show greater performance.

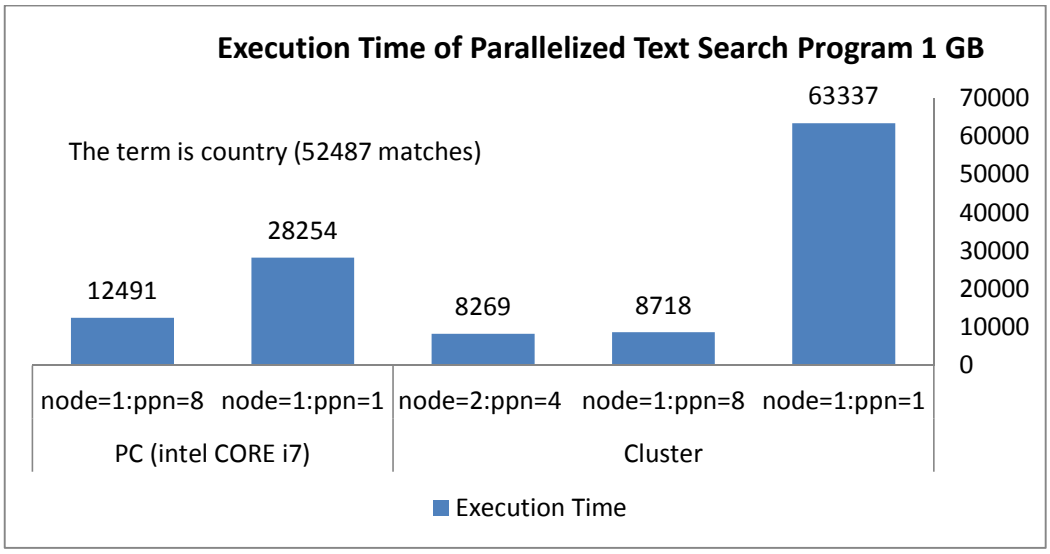

Figure 3 Execution time of search program with medium size of text

3. The performance of the application with large text files.

The information in Figure 4 demonstrates the performance of the search application running on various numbers of nodes and cores in the cluster and a PC with fairly large text file (~5 GB). Again, the given term is country and the number of matches is 264544 matches.

The performance of the application with large text files is quite similar to its performance with the medium ones. The execution time of the application on a single node with a single core in the cluster is high compare with the single-core PC. Significant improvements in performance of the search application take place when it is running on multiple cores in both the single node and the two nodes.

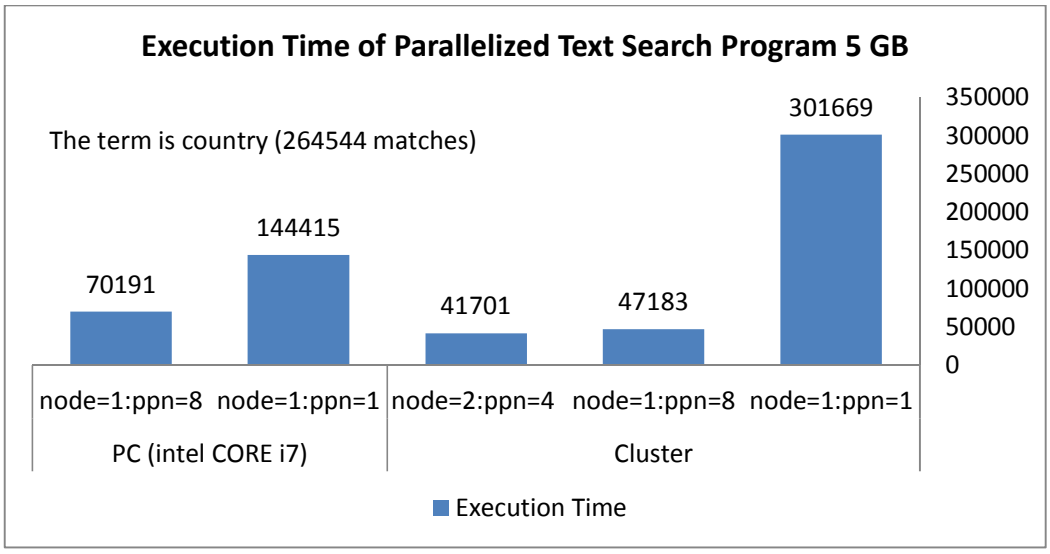

Figure 4 Execution time of search program with large size of text