Hanzehogeschool Groningen
University of Applied Sciences

Instituut voor Life Science & Technology

Bio-informatica

6-7-2018

High Performance Computing

Author:	Armin Dedic
St. number:	342615
Lecturer:	Martijn Herber
Date:	06 – 07 – 2018

Assignment 1

The first assignment is about calculating the average phred-score per base for all reads in a fastq file. To do this the ordinal values of each quality base needs to be calculated and the sum of that will be divided by the amount of bases that occurs to get the average score. This is assomplished by writing a Python version 3.5.3 script based up on a multiprocessing concept. The logic behind this idea is to run the program simultaneously on multiple cores, thus each core will process its own part of the file. The amount of cores will be provided by the user on the command-line as the third argument. When the number of cores is given the file will be divided into that amount of parts to acomplish the idea which is just mentioned.

Results

The results of the first assignment are not too bad, there is always room for improvement. To keep in mind this processing is now done on one node divided by the amount of cores that the user provides. The more cores are provided the faster it processes the file. To improve this processing time, more nodes should be used to divide the workload.

Assignment 2

The second assignment is doing the same calculation as in the previous one. The difference is that this time the workload is spread over four machines which were accessible through a ssh connection. To do this two Python scripts were created, one for the server manager and the other one was the client who connects to that server and calculates the scores. The principle is that each machine (client) will process one fourth of the file and this time each on several nodes. When all clients are finished with their job all the results are then returned to the server manager which merges it together into a list and presents it to the user.

Results

The results now are like expected much better, because each process is executed on a different machine. Each client processes its own part of the file and returns its result to the server manager. However, even this could be more faster so there is still room for improvement.

Assignment 3

For the third assignment a bash.sh script is used to call the Python script that was created in the first assignment. By providing this bash script with the amount of cpus per task and memory size it will execute the program on the perigrine cluster.

Results

The differences when comparing this assignment with the first one, is the processing time, now it has almost the same amount of improvement as assignment two has. Again with no supprise, this could only happened if the processes are executed on a higher memory size to perform the tasks. These results do not differ that much from the first assignment but there is some improvement because now it is executed from a cluster which is set up with a few parameters to speed up the process. This is also done on one node.

Assignment 4

This was the last practical assignment which I did not manage to complete. I do understand the idea and that is what I would like to explain here. It is stil the same multiprocessing concept but now applied on a peer to peer concenpt. The same way assignment three was using the Python script of the first assignment, now assignment two should be used to continue. The peer to peer concept does not have a separate client and server but instead there is a single application that acts both as client and server. In a sense, the application acts as a client, because it reaches out to a server and it acts as a server, because it accepts commands from a client. The way this should be established is by broadcasting messages to all clients available using sockets. A message like: “Hello everybody I am here with this IP addres where are you?”. By the time all clients are present and found they will randomly process a part of the file by choosing it itself. They will register each part to known which is complete and when its all finished the results are then returned.

Results

As mentioned above, this assignment was not completed and so no results are available. However, I will like to give my estimation of how this aproach would succeed. Running this script on the perigrine cluster with some powerfull parameters like a high memory size and choosing multiple nodes would perform pretty good. It would be much faster when comparing it to the previous assignments.

Spark

Spark is an Apache open source software which can be used in data analytics cluster computing system for large scale data processing. Spark is most suitable for doing batch based processing as well as real-time processing. In terms of programming languages, it is supported by Java, Scale, Python and R. It is suitable for machine learning algorithms, as it allows programs to load data into the memory of a cluster and query data repeatedly. This phenomenom is considered the future when working on project based on a parallel processing, because it takes a lot of work out of our hands by doing the most important task automatic.

Spark has various components such as:

- Spark SQL and DataFragmes
- Spark Streaming
- Machine Learning (Mllib)
- GraphX (graph)

Spark SQL – this component allows you to query your structured data inside spark programs using either SQL or the above mentioned programming languages.

Spark Streaming – has the fast scheduling capability of Spark for streaming analytics. It allows data engineers and data scientist to process real-time data from various sources like Amazon. Its key abstraction is Dstream which is a stream of data divided into small batches. Dstreams are built on RDDs which is Spark's core data abstraction. This allows Spark Streaming to integrate with any other components like Spark SQL or Mllib. The advantage of this is that it is live and you can monitor every single click on a website. Each click on a node is processed and deleted by Spark.

Machine Learning (Mllib) – is Spark's component for machine learning that consist of learning algorithms and utilities like classification, regression and clustering.

GraphX – Is a distributed graph processing framework. For the computation of graphs it provides an API and a optimized runtime for the Pregel abstraction. Pregel is a system for large scale graph processing.

Implementing our case in SPARK

Spark is considered the future when working on a parallel project. Using my understanding of how Spark works, this would not be very helpful to perform the average phred-score caulcation. The dataset we use consist of approximately 3.5 GB therefore it would be a lot of overheat when using the map and reduce methods. In our case the mapper function will map the reads with a method that the user provides on the nodes and the reduce function will merge it all together and return it back to the user. Thus, each node will only see a small amount of data. The nodes will not communicate with each other. They only known that they have received a amount of data and the method which they should perform on it. This means when using a cluster with machines like the perigrine cluster, there is no information needed from the user to think of how the data will be assigned to what node. The only thing that the user will need to do is to tell the system to use the mapper function on that data and when its done to use the reduce function to return the results.

During this course I realized how difficult it is to distributing jobs manually. After reading some papers about Spark, I could understand why it is considered to be the future when working on parallel processing projects. Spark takes the responsibility of a lot of tasks which are usually hard for the user to program it. I found it interesting and very educative to work on those assignments, because parallel processing is very useful in projects that takes a large datasets. I do regret that I did not manage to complete the fourth assignment which would be very interesting to compare the actual results.