Universidade Estadual de Campinas
Instituto de Computação

MO644 - Introdução à Programação Paralela

# Tarefa 12

*Student:*

192744, Miguel Antonio Rodriguez Santander

*Teacher:*

Guido Araújo

junho de 2017

## 1 Introduction

This laboratory consisted in working with The OmpCloud tool, which , allows to mix the tools of OpenMP with Spark and the use of clusters in the cloud. This tool, allows to solve problems using distribuited clusters, very fast and efficient. For this we prepare a pair of experiments with the multiplication of two matrices, which have very large sizes, and the calculation becomes very expensive for even a simple computer.

## 2 Programming and configuration

In order to use OmpCloud[1] in our code, it is necessary to use the OpenMP clauses that this tool has included. As we can see in the Listing 1, the first action was to map all variables that will be used in the cloud. Then for each *for* statement a parallelization is performed using the pragma of the *for* statement, and within each for a unique OmpCloud statement is used, which allows data partitions to be partitioned in the map clauses. Whit this the cloud is responsible for making the calculation using Spark, which can be hosted inside the same computer where it is hosting, or can also be in some cloud like Microsoft Azure,[2] or Amazon Web Services.[3].

```
// Kernel to parallelize using OmpCloud
void mm2_OMP(float *A, float *B, float *C, float *D, float *E) {

  //Maping data to cloud
  #pragma omp target map(alloc: C[:N*N]) map(to: A[:N*N], B[:N*N], D[:N*N]) map(
    from: E[:N*N])  device(CLOUD)
  {
    //Paralleling for
    #pragma omp parallel for
    for (int i = 0; i < N; i++) {
      //Using clausule to split data into CLOUD.
      #pragma omp target data map(to: A[i*N:(i+1)*N]) map(from: C[i*N:(i+1)*N])
      for (int j = 0; j < N; j++) {
        C[i * N + j] = 0.0;
        for (int k = 0; k < N; ++k) {
          C[i * N + j] += A[i * N + k] * B[k * N + j];
        }
      }
    }

```

[1]https://ompcloud.github.io/
[2]https://azure.microsoft.com
[3]https://aws.amazon.com/

```
    //Paralleling for
    #pragma omp parallel for
    for (int i = 0; i < N; i++) {
      //Using clausule to split data into CLOUD.
      #pragma omp target data map(to: C[i*N:(i+1)*N]) map(from: E[i*N:(i+1)*N])
      for (int j = 0; j < N; j++) {
        E[i * N + j] = 0.0;
        for (int k = 0; k < N; ++k) {
          E[i * N + j] += C[i * N + k] * D[k * N + j];
        }
      }
    }
  }
}
```

Listing 1: Function used to calculate Matrix Multiplication with clausules of OmpCloud.

For the case of this laboratory, the Microsoft Azure cloud tool was used, in which it was possible to create a cluster with Spark 2.0.1, which is named *cluster192744-3*, it contains 40 cores, the which are dividen into 8 cores divided into two nodes, as Heads or masters. The other 32 cores in 4 nodes, named workers o slaves. In order to make the connection to this cluster, OpmCloud allows the configuration through a file shown in the Listing 2.

```
[AzureProvider]
Cluster=cluster192744-3
StorageAccount=storages192744
StorageAccessKey=lX96y0fr2s/fmmU76KfK/r2OHUgNYJmkcwZXW0anuHhcEwbz+TAOQJXCPUxSpdJ/CS6dWalBl4sRsZcXEzYzow==

[Spark]
User=sshuser
WorkingDir=/home/sshuser/
```

Listing 2: Configuration for contect to Azure.

## 3 Experiments

To see how much better a cluster's performance than the serial version is, we set out to experiment with different sizes of matrices (100, 500, 1000, 1500, 2000). In Fig. 1 we can see that as the value of $N$ increases, the speedup observed by the execution in the cloud is better, because when the value is very small, the overhead cost to send the data to the cluster, running it in different machines, is greater than executing the code in local machine.

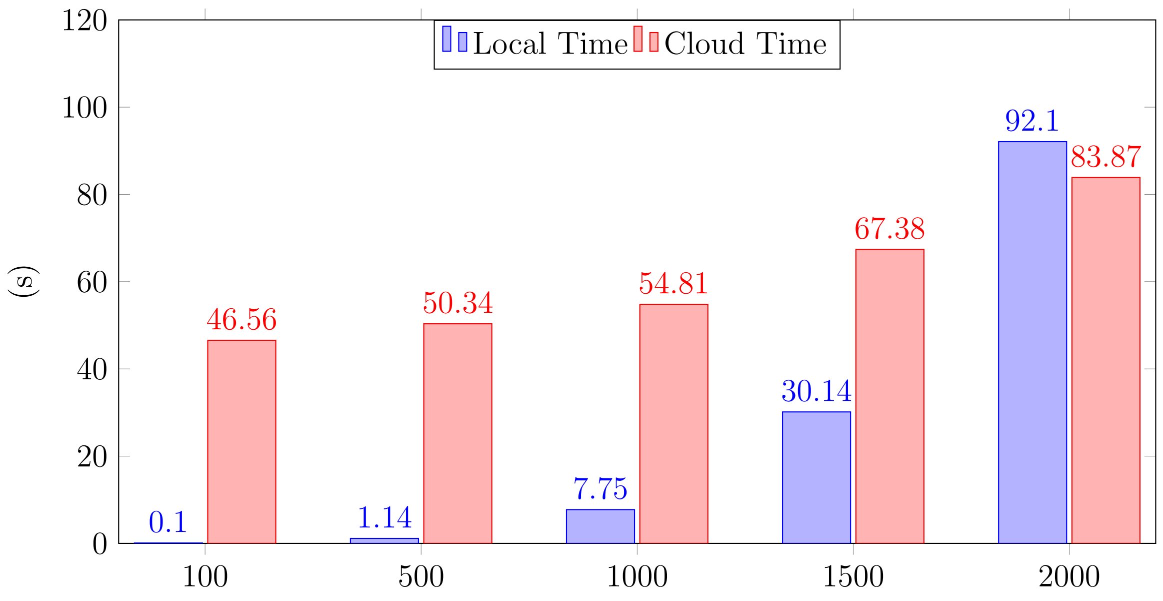
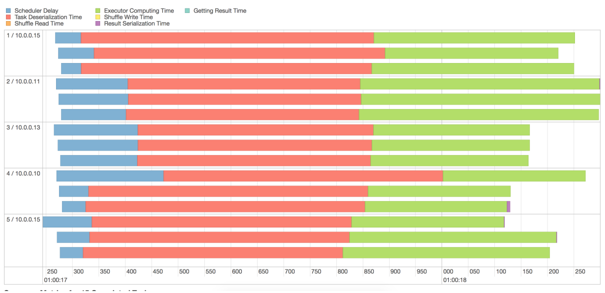


Figura 1: Speedup with different sizes of the matrix.

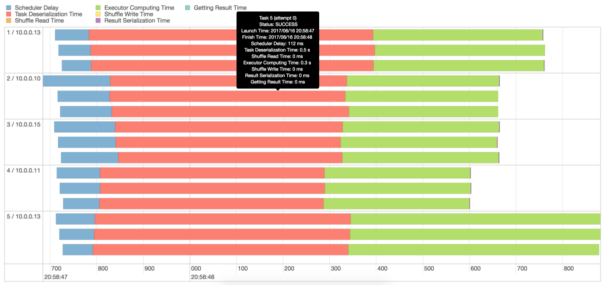
In Fig 2 we ca observe the metrics obtained by the execution of each of the experiments, where it can be observed in detail that as the value of $N$ increases, the amount of time spent in the execution is greater for each core.

## 4 Conclusion

OmpCloud is a tool that is very useful for making distributed calculation algorithms without the need to implement a complex system, so any programmer can do it. To use this tool it is necessary to take into account that the number of the calculations to be made must be large, but the resources spent on it, will be spent in vain, this because it can be better to solve the problem in local if the amount of data is low.

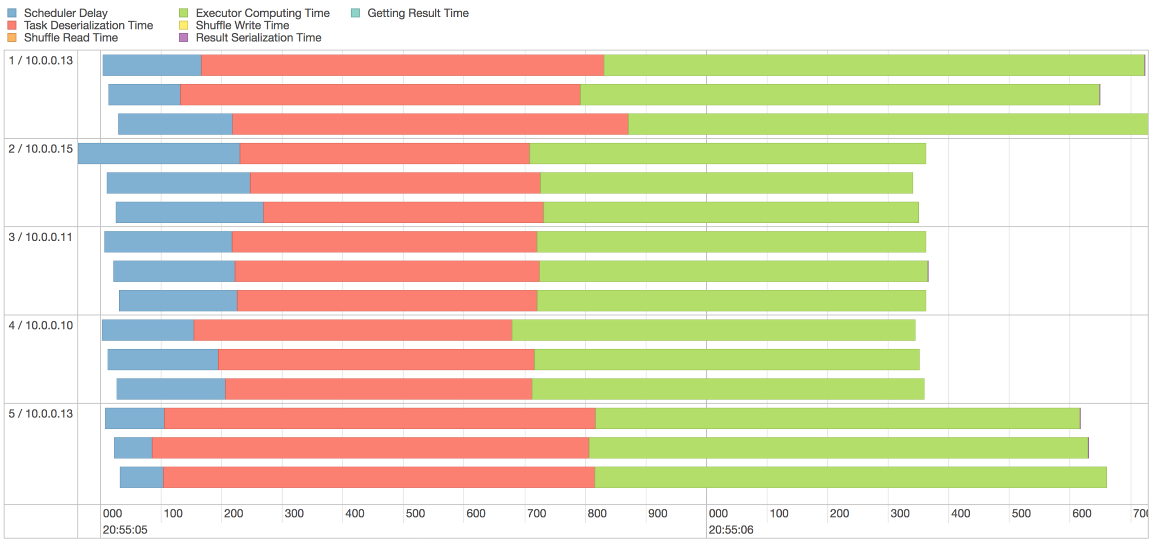


(a) Metric with $N = 100$

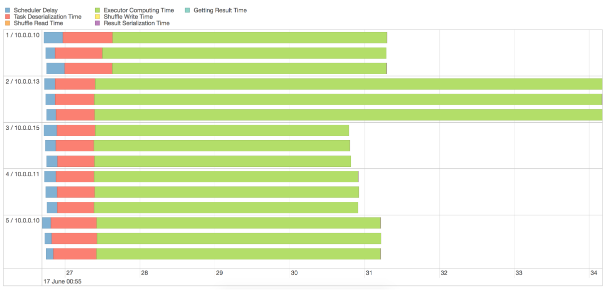


Task 5 (attempt 0)
Status: SUCCESS
Launch Time: 2017/06/16 20:58:47
Finish Time: 2017/06/16 20:58:48
Scheduler Delay: 112 ms
Task Deserialization Time: 0.5 s
Shuffle Read Time: 0 ms
Executor Computing Time: 0.3 s
Shuffle Write Time: 0 ms
Result Serialization Time: 0 ms
Getting Result Time: 0 ms

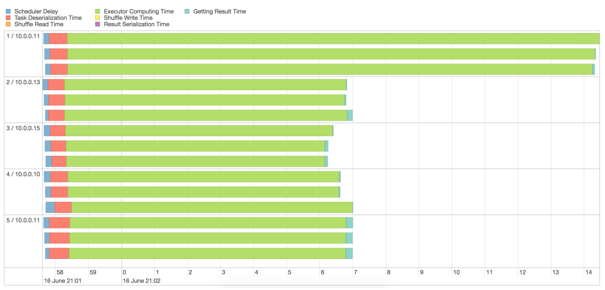
(b) Metric with $N = 500$



(c) Metric with $N = 100$



(d) Metric with $N = 1500$



(e) Metric with $N = 2000$

Figura 2: All metrics to execute into Spark.