CleanCity Computing - Internal Report

Marko Niinimaki*, Dimitri Racordon*, Nabil Abdennadher and Roger Schaer**

*HES-SO, Geneva
**HES-SO Valais

June 2013

1. Abstract

This report describes the alternatives of distributed computing that were tested for a large simulation in the CleanCity project. The alternatives were local clusters based on OpenStack[1], Amazon cloud, and CloudBroker portal. We evaluate the setting up and executing a 5GB model of air currents in the “Quartier des Banques” quarter of Geneva using the OpenFOAM [2]software. The purpose of this report is to give guidelines for future execution of similar tasks.

2. Introduction

In 2012 HES-SO launched the CleanCity project[3] aiming at offering urban engineers a decision-making tool to assess the air quality and especially pollutant dispersion. In order to estimate the effects of wind, traffic and other factors, models of interesting areas are needed, as well as software tools to simulate those effects.

The "quartier des banques" is an area in Geneva's city center, south of river Rhone. A model of the area's streets and buildings was created for the project by Tamer Mohamed-Nour, during his Master degree thesis in 2012.

1 http://www.openstack.org
2 http://www.openfoam.com
3 http://iig.hevs.ch/switzerland/clean-city.html

Figure 1: Quartier des Banques

OpenFOAM is popular open source software package for computational fluid dynamics modelling. Using the software, one can simulate scenarios of currents of gases and liquids, in our case pollution particles in the air of Quartier des Banques. Since a large model requires lots of computing power, it is customary to split the model into parts. Each part is then allocated to a processor and the processors collaborate during the simulation by using the message-passing interface (MPI). Figure 2 shows the source code used to decompose the model into 16 parts.

```
decomposePar
mpirun -np 16 simpleFoam -parallel
```

Figure 2: Splitting the model and running a simulation

There, `decomposePar` splits the model into parts based on the configuration file `system/decomposeParDict`. The next line defines that the MPI-enabled program `simpleFoam` is executed as sixteen processes. Both the splitting (`decomposePar`) and simulation (`simpleFoam`) are relatively memory-hungry. The splitting typically requires 7 to 10 GB of memory and each of the simpleFoam processes about 5GB.

3. The execution environments

Local clusters: We started our testing by running the simulation in two clusters: the *Gordias* cluster at hepia (haute école du paysage d'ingénierie et d'architecture) and the *Fast* cluster at hevs/Sierre. These clusters have rather different setups, *Gordias* has 224 CPU cores as 28 8-core nodes, each with 8GB memory. *Fast* has 24 cores in a single node with 100 GB memory. The setup favours the Fast cluster, since it can run all the processes within one node, eliminating the need for network communication between nodes. decomposePar benefits from large memory as well.

Due to the node setup of *Gordias*, we need to define the execution more carefully than in the Fast cluster where we could use the commands of Figure 2. Specifically, in Gordias we need to avoid allocating all the processes in just one node, since this would exhaust the node memory.

A command file for achieving this by using the Sun Grid Engine[4] cluster software is shown in Figure 3.

```
#$ -cwd
#$ -S /bin/bash
#$ -pe mpich 128
source ~/OpenFOAM/OpenFOAM-2.1.x/etc/bashrc
decomposePar
export PATH=$PATH:/cm/shared/apps/openmpi/open64/64/1.3.3/bin
mpirun -np 16 -npernode 1 simpleFoam -parallel
```

Figure 3: Sun Grid Engine command file

There, we "overbook" the task with the "`$ -pe mpich 128`" instruction to make sure we have at least 16 8-core nodes for the job. Then we use the "`-npernode 1`" instruction to limit the number of processes per node to one.

Amazon EC: Execution of the case on the Amazon cloud was done by using the CloudFlu[5] software that simplifies the deployment of OpenFOAM. A single cc2.8xlarge "cluster compute" instance of Amazon was used for the test. A breakdown of the execution is shown below.

- Compressing case data & uploading it to Amazon S3 : <10min
- Starting the instance, configuring it for Cloudflu, downloading and decompressing case: <10min
- Decomposing the case : ~15min
- Solving the case : ~50min
- Reconstructing the case : ~1h
- Uploading the results : ~1h (24GB to zip and upload)
- Downloading and decompressing the results : ~20min

The total time was thus ca 4 hours and the cost of computing at Amazon was 12 USD.

We ran the experiment second time on Amazon EC2 and got slightly different values:

- Compressing case data & uploading it to Amazon S3 : <10min
- Starting the instance, configuring it for Cloudflu, downloading and decompressing case: <10min
- Decomposing the case : ~15min
- Solving the case : ~50min
- Reconstructing the case : ~20min
- Uploading the results : ~20min (24GB to zip and upload)

---

[4] Currently known as Oracle Grid Engine http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html

[5] http://sourceforge.net/apps/mediawiki/cloudflu

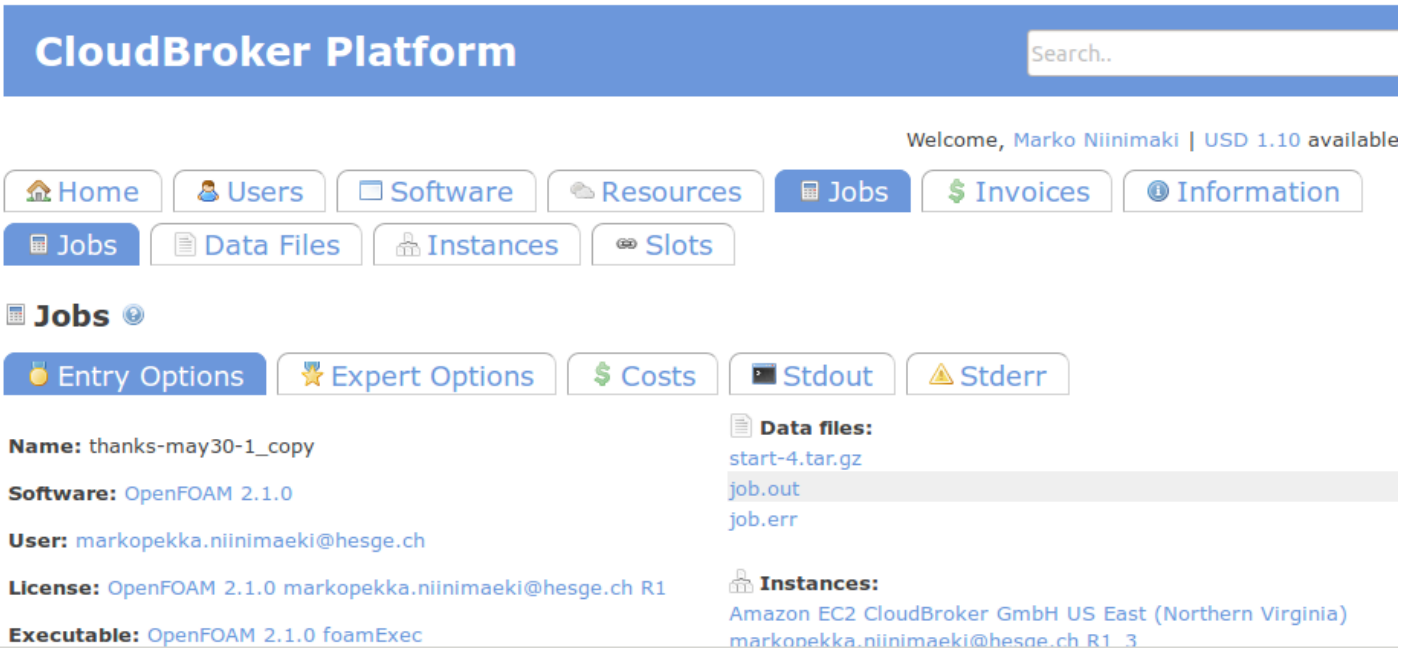
- Downloading and decompressing the results : ~30min

CloudBroker: CloudBroker[6] is a Swiss company specialised in providing scientific applications that can be executed in cloud infrastructures. OpenFOAM is one of the applications provided by CloudBroker. The portal environment with our OpenFOAM job is seen in Figure 4 and recovering an output file from the portal in Figure 5. In order to simplify data transfer, the case is a compressed tar.gz archive (1 GB). The case was executed in a quadruple extra large Amazon EC2 instance with 60 GB memory.

At the time of testing, CloudBroker's OpenFoam did not yet provide an MPI environment, so our test has been run as "simpleFoam" without parallelization. The slow execution time (120 h) and relatively high cost (285 USD, due to the cost of 2.35 per CPU hour + file transfers) reflect this.

The following steps were taken to execute the job in CloudBroker:

- A user account was created on cloudbroker.com.
- A new job "thanks-may30" was created on the portal.
- OpenFoam was selected as the job's application and `foamExec` as the jobs's executable. In the expert options `simpleFoam` was given as `foamExec`'s parameter.
- The compressed input file was created by `tar cvzf start-4.tgz`
- The file was uploaded on the thanks-may30 job's files page. In the expert options tab, it was marked that this file is an archive.
- The quadruple extra large Amazon EC2 instance was selected as this job's instance.
- The job was started.



6 http://cloudbroker.com

Figure 4: The CloudBroker portal with an OpenFOAM job.

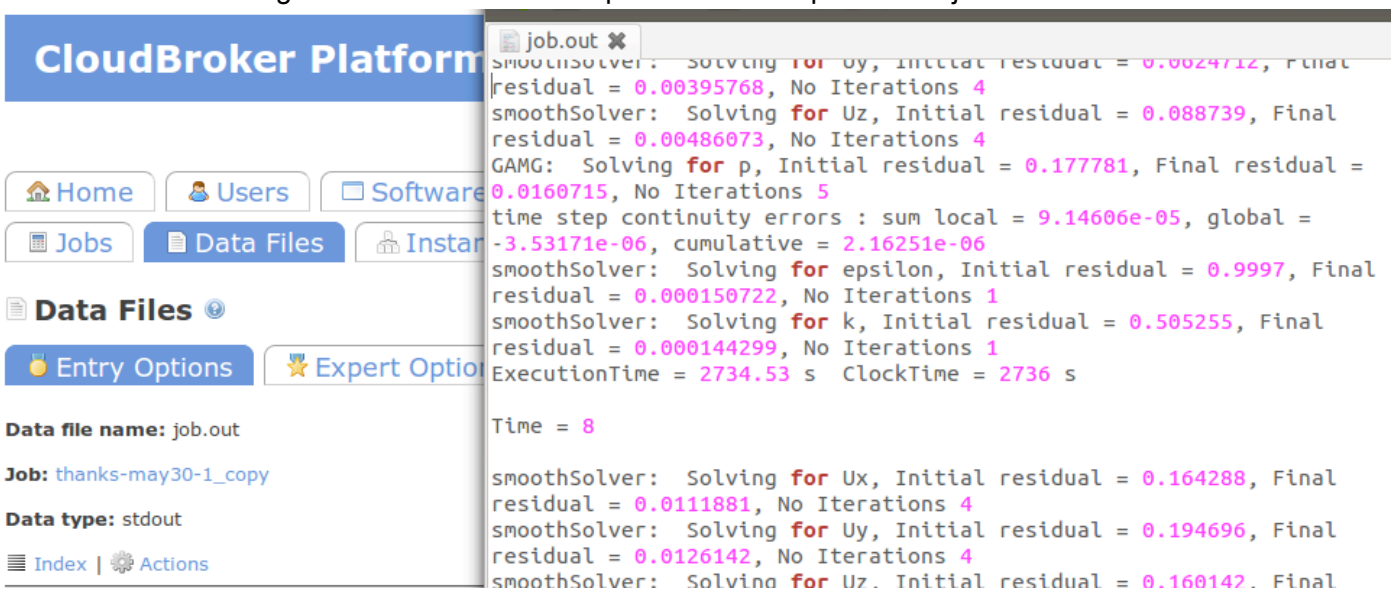


Figure 5: Checking the job's stdout

OpenStack: OpenStack is a collection of python tools to set up Cloud IaaS infrastructures. We installed our own OpenStack infrastructure, called hepiaCloud, which is spread over two types of machines. Slightly less than 40 machines run on an Intel(R) Xeon(R) E31225, with 4 cores at 3.10 GHz, and 32GB Ram. In addition, seven machines run on an Intel(R) Xeon(R) X5650, with 6 cores at 2.67 GHz, and 128GB Ram. All machines are connected on a 1Gb network.

For the sake of the experiment, we did not need to start several virtual machines, so we created one VM with 8 vCPU and 60.5 GB Ram. This flavor matches the Amazon's **cc2.8xlarge** (see http://aws.amazon.com/ec2/instance-types/). As the case requires a lot of space on the filesystem, we also mounted a 150GB volume to the VM instance, over iSCSI.

The following steps were taken to execute case job on hepiaCloud:

- Start a virtual machine with a flavor similar to the Amazon cc.8xlarge
- Upload and decompress the archive containing all data
- Decompose the case into parallel data sets
- Run the solver
- Zip and download the results

4. The results and evaluation

Table 1 shows the results of execution on the local clusters (gordias and fast.hevs.

	fast.hevs	**gordias**
12 process	11h	21h
18 process	17h	6h 6min
27 process	-	4h 10min

Regarding cloud platforms, measures are the following (times are in minutes):

- Amazon: 225 (ca 4 hours) the first time, 155 (ca 1 and half hours) the second time
- hepiaCloud: 410 (ca 7 hours)
- CloudBroker: 7224 (ca 120 hours, no parallelization)

In comparison: running simpleFoam without parallelization in fast.hevs.ch takes about 55 hours. It should be noticed that we do not include data transfer times, but they are generally not significant.

Even if we ran the same VM instance flavor on OpenStack than we did on Amazon, we measured significant differences regarding the execution times. The main reason to this gap is the lack of high availability network storage system on our own OpenStack infrastructure.

Ease of use is naturally a matter of opinion. Here we need to assume that the user generally knows how to use the OpenFOAM software but does not know anything about cloud systems (like Amazon) or portals (like CloudBroker). All the platforms proved usable. However, because of its “simplicity” the Fast cluster was most straightforward to use, and the CloudBroker portal was probably the most novice-friendly.