INP Grenoble – ENSIMAG
École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble

Second year internship report

LIG and UShareSoft

Using OAR for ISO Image Creation

LAHIA M'hammed
2^{nd} year

23 June 2008 – 17 September 2008

Laboratoire LIG - ENSIMAG
Antenne de Montbonnot
ZIRST 51, avenue Jean Kuntzmann
38330 MONTBONNOT SAINT MARTIN

Tutor
Grégory Mounié

UShareSoft, SAS
10 bis rue Ampère
38000 Grenoble

Team
Thomas Debru, James Weir
Olivier Bourdon

Contents

1 Introduction

I spent the first half time of my internship in the LIG laboratory and the second half UShareSoft office in Grenoble, I collaborated with the UShareSoft Engineering team, supervised by Mr. Thomas DEBRU.

1.1 The UShareSoft Team

UShareSoft provides one of the first on-demand Software as a Service (SaaS) platforms enabling communities and developers to create and deliver open source software appliances or virtual appliances. This is a breakthrough concept, allowing solution providers to be more successful by letting them integrate their software into automated open source software solutions in record time. Primary target customers for UShareSoft are, Independent Software Vendors, System Integrators and Solution Providers. UShareSoft's platform targets a mix of technical profiles including developers, system administrators, technical pre-sales as well as software delivery and integration engineers. Software solutions or software appliances are applicable to any software workload running on a desktop to a x86/x64 server system, with the capabilities to address both simple and complex software stacks.

The team I worked with

Thomas DEBRU	VP Engineer
James WEIR	CTO
Olivier BOURDON	PDG / CEO

1.2 The LIG Team

LIG Laboratoire d'Informatique de Grenoble, created in January 2007, it gather about 500 researchers, doctors and technicians, forming 24 research team. LIG s academic partners are : CNRS, Grenoble INP, INRIA, UJF and UPMF.

The team I belong to is the MOAIS team

MOAIS focuses on the programming of parallel applications where increasing the number of resources is a key to improve performance: beyond the optimization of the application itself, the effective use of a larger number of resources is expected to enhance the performance. This encompasses large scale scientific interactive simulations (e.g. immersive virtual reality) that involve various resources: input (sensors, cameras, ...), computing units (processors, memory), output (video projectors, images wall) that play a prominent role in the development of high performance parallel computing. The research axis of the MOAIS team are focused on the scheduling problem with a multi-criteria performance objective: computing efficiency, precision and reactivity. The originality of the MOAIS approach is to use the application's adaptability to enable its control by the scheduling. The critical points concern designing adaptive malleable algorithms and coupling the various components of the application to reach interactivity with performance guarantees. MOAIS is a partner of the network of excellence CoreGRID, the French project Grid'5000 and the experimental platform GrImage. International academic collaborations include Idaho State University (USA), Universidade Federal do Rio Grande do Sul (Porto Alegre, Brazil) and Universidade de SÃ£o Paulo (Sao PÃ£ulo, Brazil). Among the industrial partners are STMicroelectronics and Bull companies.[1]

The technology I was asked to use was OAR. This is a batch scheduler developed by LIG. In order to configure and integrate OAR into UShareSoft's infrastructure I had to collaborate with both the LIG and UShareSoft teams (consisting of researchers and engineers). My primary contacts were Dr. Grégory MOUNIÉ and Dr. Olivier RICHARD.

1.3 OAR

1.3.1 Definition & Architecture

In OARADMIN documentation Oar is an opensource batch scheduler which provides a simple and flexible exploitation of a cluster. It manages resources of clusters as a traditional batch scheduler (as PBS[2] / Torque[3] / LSF[4] / SGE[5]) In other words, it doesn't execute your job on the resources but manages them (reservation, access granting) in order to allow you to connect these resources and use them.

OAR is used by Grid'5000 which is a french national project aiming to put in use an experimental grid of 5000 processors. There are 9 participant sites: INRIA Sophia, IRISA

[1]From LIG Website: http://www.liglab.fr/

[2]Portable Batch System (or simply PBS) is the name of computer software that performs job scheduling. Its primary task is to allocate computational tasks, i.e., batch jobs, among the available computing resources. PBS was originally developed by MRJ for NASA in the early to mid-1990s. MRJ was taken over by Veridian, which was later taken over by Altair Engineering, which currently distributes PBS Pro commercially and partially maintains an Open Source version (OpenPBS). [From Wikipedia]

[3]TORQUE (Tera-scale Open-source Resource and QUEue manager) is a resource manager providing control over batch jobs and distributed compute nodes. Torque is based on OpenPBS version 2.3.12 and incorporates scalability, fault tolerance, and feature extension patches provided by USC, NCSA, OSC, the U.S. Dept of Energy, Sandia, PNNL, U of Buffalo, TeraGrid, and many other leading edge HPC organizations.

[4]Platform LSF is software for managing and accelerating batch workload processing for compute-and data-intensive applications.

[5]Sun Grid Engine (SGE), previously known as CODINE (COmputing in DIstributed Networked Environments) or GRD (Global Resource Director),[1] is an open source batch-queuing system, supported by Sun Microsystems. [From Wikipedia]

Rennes, Loria Nancy, LIP Lyon, Toulouse, LABRI Bordeaux, LIFL Lille, ID Grenoble and LRI ORSAY.

To install the OAR batch scheduler, you require administrator privileges of the cluster. The Grid architecture consists of three distinct parts, namely:

- A Server machine which runs the OAR engine. In this machine the server package must be installed and all the grid nodes must be registered in the OAR database *(I used Mysql for this work)*, with all the desired properties with appropriate configuration.
- A front-end machine from which we can launch OAR commands for task execution.
- Computing nodes where all the jobs submitted from the front-end machine will be executed. During the configuration it is imperative that an OAR user is created on each of the computing nodes. This guarantees that all ssh connections between the nodes will be successful without requiring a password.

To handle job execution in the configured grid, OAR uses:

- A database where all the jobs, resources, scheduling, history and logs are stored (see figure 1).
- A set of queues to define job priority.

1.3.2 OAR Server Modules

1. **Almighty:** The OAR server, managing the jobs to be scheduled (see figure 2).
2. **Sarko:** Module that is executed periodically by Almighty to detect and kill expired jobs.
3. **Judas:** Module providing the printing and logging service.
4. **Leon:** Module that handles the killing of jobs when asked by other modules or commands.
5. **Runner:** Module that launches the jobs in the allocated nodes.
6. **NodeChangeState:** Module used to change the resources state and to check on the executing jobs.
7. **Scheduler:** Module to check for valid reserved jobs, launch them on the right resource and calculate the estimated time to execute for jobs still in the queue.

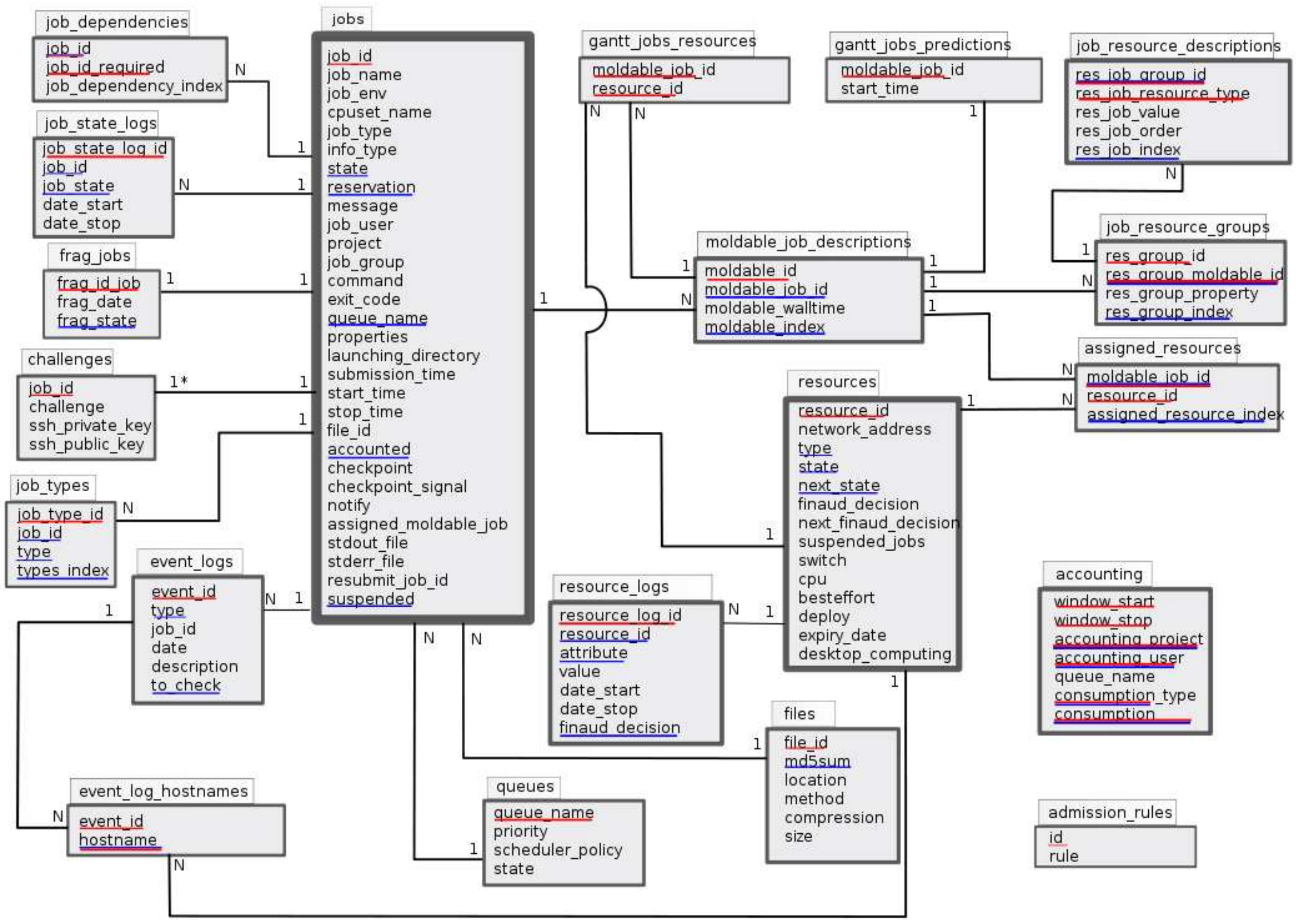
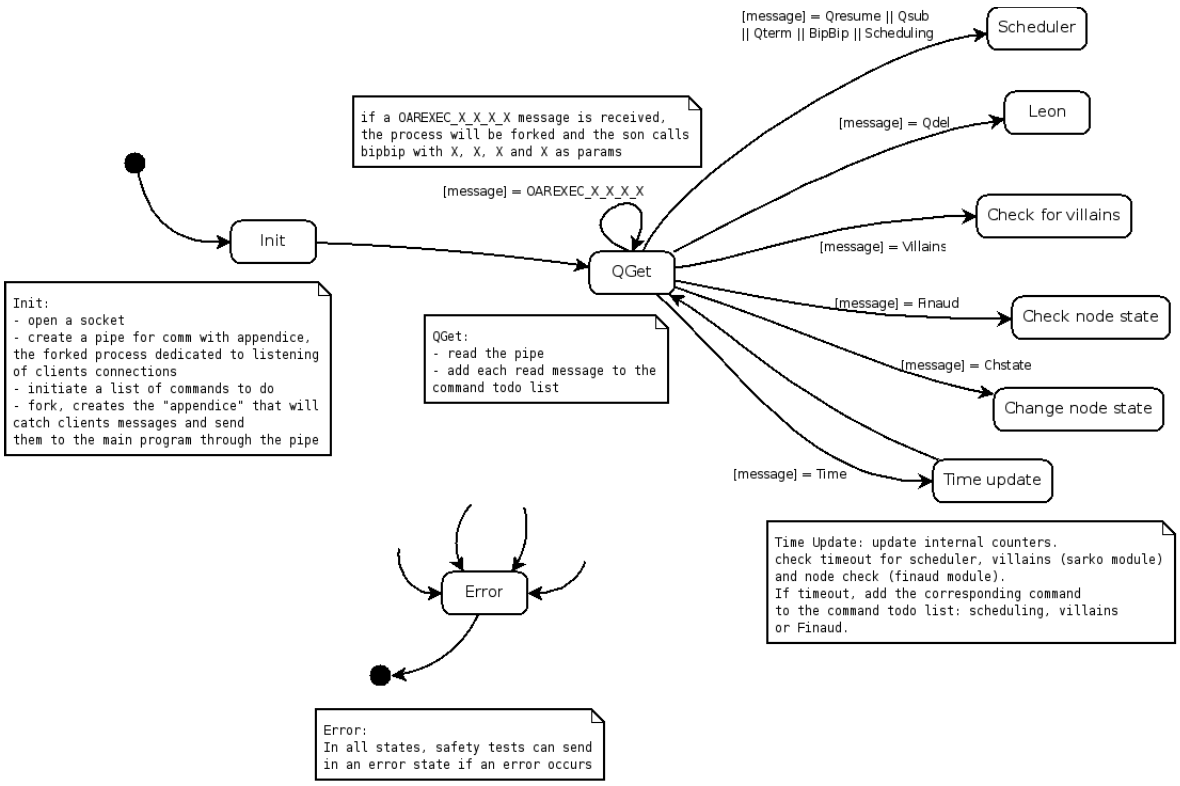


Figure 1: Database schema (underlined field is PRIMARY KEY or INDEX)



```
if a OAREXEC_X_X_X_X message is received,
the process will be forked and the son calls
bipbip with X, X, X and X as params
```

```
Init:
- open a socket
- create a pipe for comm with appendice,
the forked process dedicated to listening
of clients connections
- initiate a list of commands to do
- fork, creates the "appendice" that will
catch clients messages and send
them to the main program through the pipe
```

```
QGet:
- read the pipe
- add each read message to the
command todo list
```

```
Time Update: update internal counters.
check timeout for scheduler, villains (sarko module)
and node check (finaud module).
If timeout, add the corresponding command
to the command todo list: scheduling, villains
or Finaud.
```

```
Error:
In all states, safety tests can send
in an error state if an error occurs
```

Figure 2: Almighty general scheme

1.4 UShareSoft Web Services

The UShareSoft Web Service is designed as an online platform that allows users to create and design their own OS images based on a set of open source projects. The platform is developed almost entirely using JAVA technologies. The platform is scheduled to be in use for the beginning of 2009.

2 Subject

2.1 Mission

During 85 days, my mission was to design and implement the integration between OAR, and the UShareSoft platform. The OAR grid server as a set of computing nodes to generate all the ISO images that have been requested by the users. This included defining a common interface between the grid and the Web Service, as well as properly configuring and tuning OAR appropriately. One of the requirements of the grid was to be able to handle multiple client profiles (paying and non-paying customers).

2.2 Architecture Overview

The architecture can be split into four distinct parts; namely:

- UShare Factory Web Service: This is the front-end web application, providing the access and exposing the services to customers. The web service is RESTful
- Knowledge database: The database holding all the catalogue information (OS distributions and projects); user information; groups and appliances
- The Grid Engine: Providing the resource pool for software appliance image creation
  - Server Node providing job scheduling.
  - One of several target nodes used to execute the jobs that generates the ISO image.
- UShare Appliance Builder: Client-side RIA (Rich Internet Application) used to connect to the back-end web services

This architecture is illustrated in figure 3.

3 Work

3.1 The Internship Progress

First 15 days: Familiarization of OAR including installation, configuration and the tools. AT the end of the two weeks I was able to install and configure the grid. This

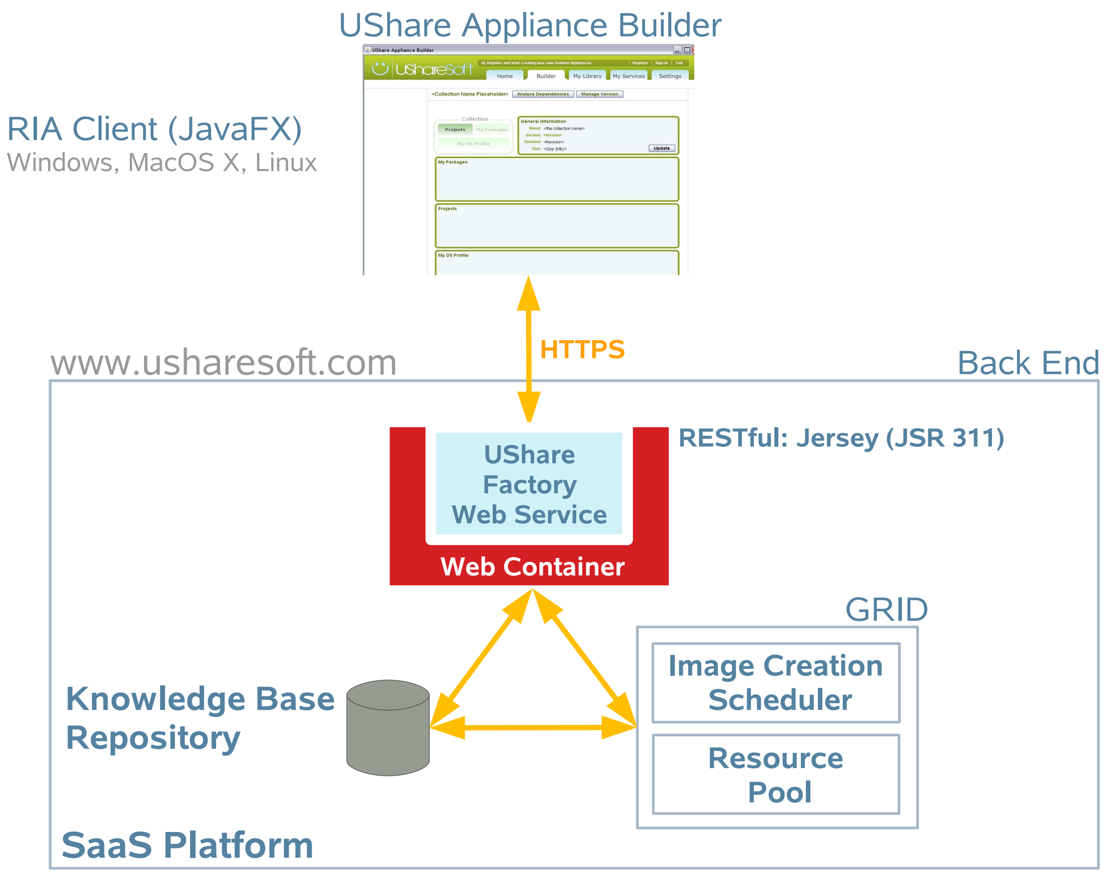


Figure 3: Architecture Overview

included verifying library dependencies. During this experience I increased my system administration skills.

Next 15 days: In this period I had to investigate how UShareSoft's Web Service worked and how I could integrate it with OAR. During this time I worked with UShareSoft's engineering team, setting up my development environment, gathering their requirements and becoming familiar with their source code. By the end of this period I managed to integrate OAR UShareSoft's Web Service, therefore when a user requested to generate an ISO image via the Web Service an OAR job was automatically created and executed by the Grid.

Last 55 days: Having a positive result during the last month, I was able to think about the way to integrate the other needs:

- Making priority between profiles using the queues in OAR
- Identifying the jobs by their "job_name" using the ticket number of the image generation request
- A real time transmission of the job's status, using the OAR database
- Detecting eventual bugs that can occurs during the job execution and notifying the client
- Creating a testsuite that can be launched during regression testing. The testsuite included functional, robustness and stress tests.

The last two weeks of this period was spent writing this report.

3.2 Technical Skills Used

For OAR: I increased my UNIX administration skills. these included :

- Manipulating system users and groups, the sudoers file and system variables
- Configuring a ssh server
- Configuring the mysql server, (database creating, permissions etc)

For the UShareSoft code: I had to update their Java code base and discovered new technologies related to JAVA:

- The use of Netbeans[6] for java programming.

[6]NetBeans refers to both a platform for the development of Java desktop applications, and an integrated development environment (IDE) developed using the NetBeans Platform.

- Manipulating XML files
- The JAVA-FX[7] technology
- Installing and configuring a GlassFish[8] server and deploying a web application.

3.3 Difficulties & Solutions

During the design and implementation of the solution there were several problems I had to solve. These are :

- Generation of scripts that provided a status notification
  - As a first thought, using the output files produced by OAR for every job, was the key to that problem, and the simplest way to resolve it is parsing these files from the JAVA code. But after having a long discussion with Mr. Thomas DEBRU and Mr. James WEIR, due to performance problems that it brings another way was required. I decided to use the OAR database to store the job's status. I wasn't sure about the way to do it, so it took me about four days to implement and test the notification scripts, the first was a Perl script and the second a small bash script.
- Integration of the produced work into the JAVA code
  - Now I had the complete procedure, from launching the job to downloading the final image. At this point I was ready to integrate my work into the main project. Prior to this, UShareSoft had a specific requirement, namely to allow the possibility to run the infrastructure with or without the grid by changing a parameters in their configuration file. The reason was to allow UShareSoft engineers to easily continue their development without the need to have a grid, and then be able to do a complete test cycle with a grid during pre-production. This kind of manipulation demanded a complete mastery of the JAVA code that I was about to modify. It took less than a week to realize and test the complete process.
- Preparing an automated way to install and configure the OAR server
  - To do so, I had to make a review of all the libraries I used to install OAR. I had also to validate the installation procedure in the Fedora OS which is different from the Debian OS where OAR is mostly used. After that I had to make a bash script to emulate the pre-configuration part, the installation part and a basic post-configuration step. After the validation of this part, Mr. Thomas DEBRU

[7]It's a new API (Application Programming Interface) for interface designers developed by Sun Microsystems

[8]An application server project by Sun Microsystems for the Java Enterprise Edition (Java EE) platform.

used the produced script to make an rpm file where he put all the tools that the web sever will use, resolving of course all the library dependencies. Then with the UShareSoft's online platform I generated an ISO image containing all of UShareSoft's infrastructure and OAR.

3.4 Tests

It is necessary to carry out a complete testing phase in a pre-production environment prior to rolling it out into production. The testing campaign requires to have a testsuite that contains functional, robustness and stress tests to emulate a real scenario when the infrastructure will be in production where 1000's of users may be using the service simultaneously. It was my task to write a tool to simulate users using the platform simultaneously, with the aim of testing the infrastructures limits.

I wrote the tool in Java and Swing. I also used UShareSoft's client API to generate the appropriate HTTP requests to the Web Service (see figure 4)

To automatically generate an ISO image, the following steps are requested :

1. Create a new user account
2. Create an appliance by choosing an OS profile and some projects
3. Request to generate an ISO image of the appliance

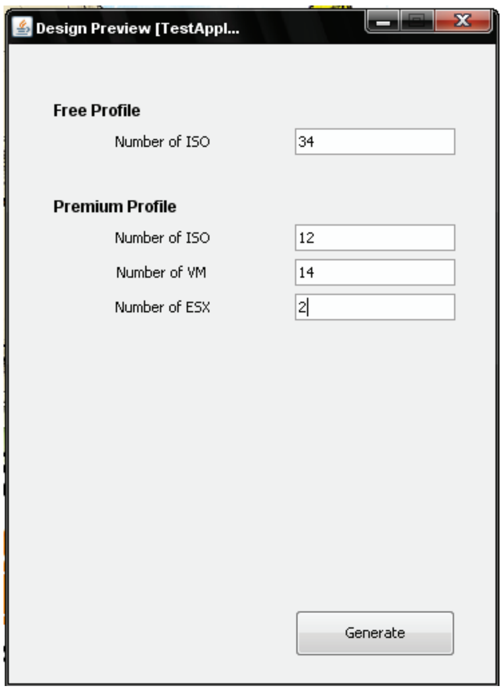
Tests characteristics & Results

- The target machine is a *Sun Ultra 20*:
  - **Processor**: Dual-core AMD Opteron 1200 series CPU with 8-GBps HyperTransport.
  - **Secondary cache**: 1MB per core.
  - **Memory**: 2GB of DDR2-667 registered ECC[9] memory.
  - **Networking**: Two Gigabit Ethernet ports.
  - **I/O**: Six USB 2.0 ports, two in front and four in back Two IEEE 1394a (FireWire) ports in front.
- The launched jobs have all the same priority they were launched from the best effort queue.

[9]EEC stands for "Error Correction Codes" and is a method used to detect and correct errors introduced during storage or transmission of data. Certain kinds of RAM chips inside a computer implement this technique to correct data errors and are known as ECC Memory.

- Every job's request is launched using a new java thread.

Due to lack of machines, I wasn't able to use a real grid, so I created three virtual nodes in the same machine where I deployed the oar server and the web service.
As a result, I was able to run five jobs in the same time, three of them were running and two were waiting in the OAR queue. After one of the running jobs finished, it was replaced by another job from the waiting queue. Also for the notification part, every thread was able to receive the exact status of his corresponding job. This result can't be used for benchmarking, in fact I didn't use a real grid, but it shows that the application can handle this kind of approach when the real grid will be setup.
A single job on the server takes 9min30s to produce an iso image using the basic profile for FedoraOS core9.
In figure 5, we can see that the jobs launching time increase with the number of simultaneous requests, it is a natural result since the Web Server is handling all images profiles construction before the oar submission. Concerning the job execution time, obtained results can't be exploited due to the absence of a real grid.



Free Profile

Number of ISO	34

Premium Profile

Number of ISO	12
Number of VM	14
Number of ESX	2

Figure 4: The test application interface

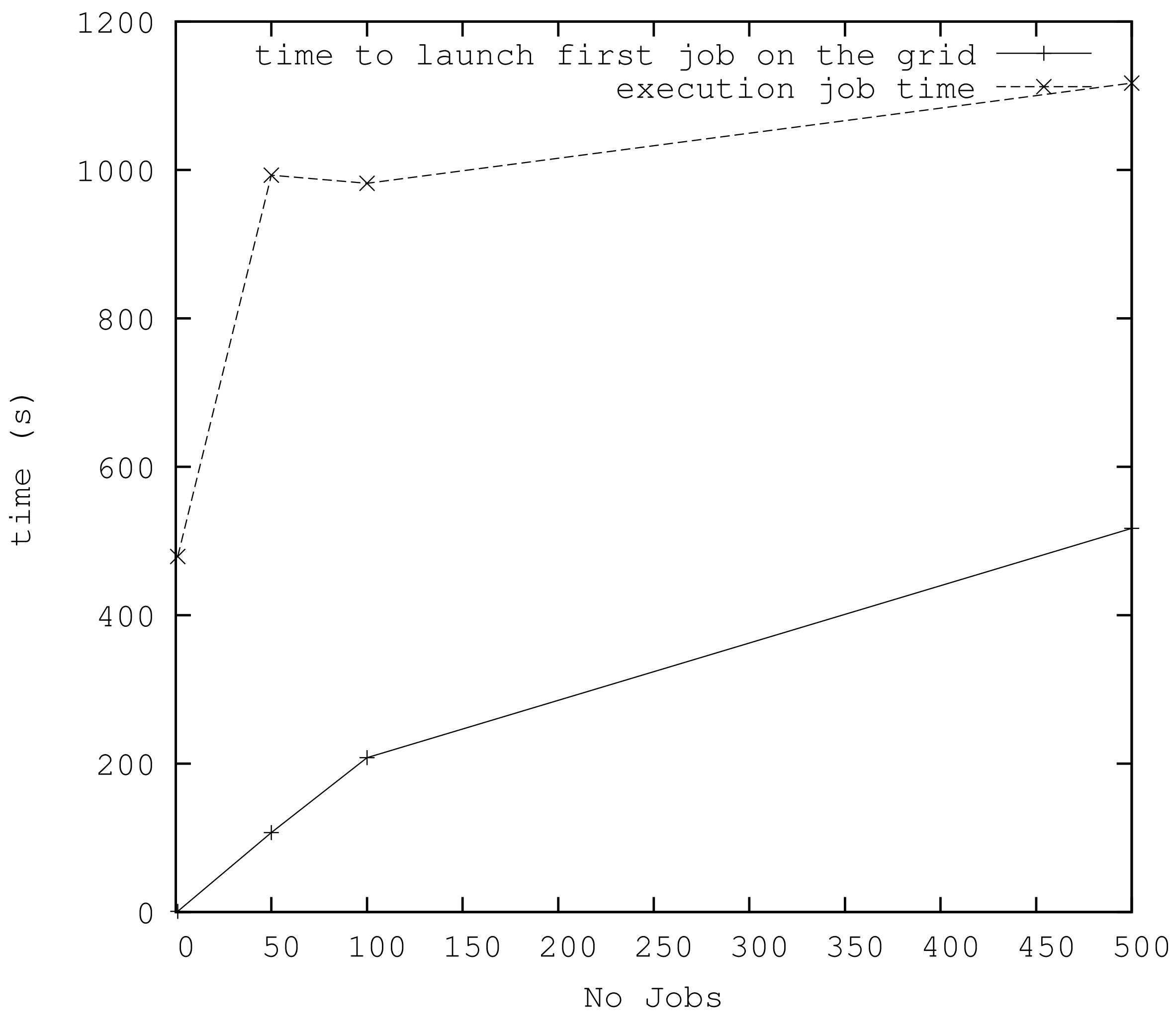


Figure 5: Test Results

4 Conclusion

As a conclusion, I can say "Mission Accomplished". During this internship I achieved new and powerful skills, It also allows me to work in a real and professional environment with real problems, real needs and with wonderful peoples from different domains. But the big work will follows, configuring a real grid with all the problems it brings, making stress tests and gathering performance statistics in order to tune the grid. It might also be interesting to modify the developed notification process using new OAR tools like *oarmonotoring* or perhaps others.

5 ANNEXES

A OAR Configuration

A.1 Basic Configuration for OAR

User configuration
the user who will launch all the oar commands.

```
useradd -m usstest
```

in the /etc/sudoers file we must add :

```
usstest ALL = NOPASSWD ALL
```

Defining the priority between profiles
to do so we will use a job's queue for every profile.

```
oarnotify add_queue "premium, 1, oar_sched_gantt_with_timesharing"
oarnotify add_queue "pro, 2, oar_sched_gantt_with_timesharing"
oarnotify add_queue "enterprise, 3, oar_sched_gantt_with_timesharing"
```

and all the free profile jobs will launch in the best-effort queue.

Creating resources

```
oarnodesetting  -a -h < nodename >
                -p network_address =< nodeipaddress >
                -p cpuset =< numcpu >
                -p core =< numcore >
```

Making job's requests

su $< profile\_login >$ -c "oarsub -d $-directory = "/home/ < profile\_login > "$
-n $< jobname >$
$-project = "customer\_login"$
-q $< profile\_queue >$
-l $/network\_address = 1/core = (2or1)$
./script.sh "

A.2 Useful OAR Commands

oarstat view the list of running and waiting jobs.

oarnodes view all the grid's nodes.

oarnotify -l view the defined priority queues.

oardel -j $< JobId >$ delete a job.

oarnodesetting -r $< resoucenumber >$ **-s** $< Alive/Dead >$ change the node's state.

oarremoveresources $< resourcenumber >$ delete a node from the grid, but the state must be changed to the "Dead" value before.

B How to Configure the OAR Server & Grid Nodes

The server due to what produced Mr. Thomas DEBRU, the server configuration is assumed to be an .rpm file.

The Grid's nodes In every node we have to repeat the following steps:

1. OAR node's pre-configuration
   - Create the oar user : useradd -m oar
   - Change the sudoers file to grant oar appropriate permissions
   - In /home/oar/.ssh/ directory copy the id_rsa.pub, authorized_keys and config files from the server machine.
   - Configure the ssh server
   - Put the UShareSoft tools in the right path for the job's executions.
2. Install oar node using the source code or the .rpm file.

The script:

```
#/bin/bash

#Adding the oar user
useradd -m oar

#preparing right prmitins for oar user

echo '#OAR configuration ' >> /etc/sudoers

echo 'Defaults>oar env_reset,env_keep += "PWD_DISPLAY_OAR_JOB_ID
OAR_JOB_KEY_FILE", !requiretty ' >> /etc/sudoers
echo 'Defaults:oar env_reset,env_keep += "DISPLAY
SSH_CLIENT_SSH2CLIENT_SHELL", !requiretty ' >> /etc/sudoers

echo 'Cmnd_Alias OARCMD = /usr/local/oar/oarnodes,
 /usr/local/oar/oarstat,\' >> /etc/sudoers
echo '/usr/local/oar/oarsub, /usr/local/oar/oardel,
 /usr/local/oar/oarhold,\' >> /etc/sudoers
echo '/usr/local/oar/oarnotify, /usr/local/oar/oarresume,
 /usr/local/oar/oarsh' >> /etc/sudoers
echo '%oar ALL=(oar) NOPASSWD: OARCMD' >> /etc/sudoers

echo 'oar ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers

echo 'Defaults:www-data   env_keep += "SCRIPT_NAME_SERVER_NAME
_SERVER_ADMIN_HTTP_CONNECTION_REQUEST_METHOD_CONTENT_LENGTH
_SCRIPT_FILENAME_SERVER_SOFTWARE_HTTP_TE_QUERY_STRING_REMOTE_PORT
_HTTP_USER_AGENT_SERVER_PORT_SERVER_SIGNATURE_REMOTE_ADDR_CONTENT_TYPE
_SERVER_PROTOCOL_PATH_REQUEST_URI_GATEWAY_INTERFACE_SERVER_ADDR
_DOCUMENT_ROOT_HTTP_HOST"' >> /etc/sudoers

echo 'www-data ALL=(oar) NOPASSWD: /usr/local/oar/oar-cgi' >> /etc/sudoers

#Configuring ssh-server to accept oar connections without password

echo '#CONFIGURTION FOR OAR' >> /etc/ssh/sshd_config
echo 'AcceptEnv OAR_CPUSET SUDO_JOB_USER' >> /etc/ssh/sshd_config
echo 'PermitUserEnvironment yes' >> /etc/ssh/sshd_config
echo 'UseLogin no' >> /etc/ssh/sshd_config
echo 'AllowUsers oar' >> /etc/ssh/sshd_config
echo 'X11Forwarding yes' >> /etc/ssh/sshd_config
echo 'X11UseLocalhost no' >> /etc/ssh/sshd_config

mkdir /home/oar/.ssh
chown oar /home/oar/.ssh
chgrp oar /home/oar/.ssh
chmod 700 /home/oar/.ssh

#Configuring ssh oar_key
cp <serevr>/home/oar/.ssh/id_rsa.pub /home/oar/.ssh/
```

```

echo 'Host *' > /home/oar/.ssh/config
echo '    ForwardX11 no' >> /home/oar/.ssh/config
echo ' StrictHostKeyChecking no' >> /home/oar/.ssh/config
echo '         PasswordAuthentication no' >> /home/oar/.ssh/config
echo '         AddressFamily inet' >> /home/oar/.ssh/config

##Giving appropriate permissions
chown oar:oar /home/oar/.ssh/id_rsa.pub
chown root:root /home/oar/.ssh/authorized_keys /home/oar/.ssh/config

chmod 644 /home/oar/.ssh/authorized_keys /home/oar/.ssh/config
 /home/oar/.ssh/id_rsa.pub

#Install

cp /usr/local/uss-factory/conf/oar-2.2.13.tar.gz /tmp
cd /tmp ; tar -xvzf oar-2.2.13.tar.gz
cd /tmp/oar-2.2.13
mkdir -p /var/lib/oar
make node-instal
cd / ; rm -Rf /tmp/oar-2.2.13

# Restart the ssh server
service sshd restart
```

C Produced Scripts

C.1 How to Update Job Status

This script parses the executed job output to update the job's status on the database.

```
#!/usr/bin/perl -w

use DBI;
use strict;

my ($line,$JOB_ID,@Words,$word,$dbh,$sth,$retour,$Lcheck);

$Lcheck = "";
$JOB_ID = $ARGV[0];
$dbh = DBI->connect(
            'dbi:mysql:database=oar;host=uss;port=3306',
            'oar',
            'oar',
            {RaiseError=>1,AutoCommit=>0}
            );

while( defined( $line = <STDIN> ) )
```

```
{
    print $line;
    chomp $line;
    if ( $line =~ m/(\[\%\s*\d+\,(\s\w+)*\])/ )
    {
        $Lcheck=$line;
        $sth = $dbh->prepare(
                    "UPDATE␣jobs
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣SET␣project='$line'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣WHERE␣job_id=$JOB_ID"
                    );
        $sth->execute();
        $dbh->commit();
    }
}
if ( $Lcheck =~ m/(\[\%100, Done\])/ ){
    $dbh->disconnect();
    exit 0;
}else{
    $sth = $dbh->prepare(
                "UPDATE␣jobs
␣␣␣␣␣␣␣␣␣␣␣␣SET␣project='[%0,␣Failed]'
␣␣␣␣␣␣␣␣␣␣␣␣WHERE␣job_id=$JOB_ID"
                );
          $sth->execute();
          $dbh->commit();
    $dbh->disconnect();
    exit 1;
}
```

C.2 How to Read the Job Status

This command is what's used to notify clients on their job's status.

```
#!/bin/sh

echo "Select␣project␣from␣jobs␣where␣job_name=$1␣order␣by␣start_time"
      | mysql -uoar oar -poar
```

C.3 How to Automate the OAR Configuration

The script used in the .rpm file to configure the OAR part of the UShareSoft server.

```
#/bin/bash

#Adding the oar user
useradd -m oar

#preparing right prmitins for oar user
```

```

echo '#OAR configuration ' >> /etc/sudoers

echo 'Defaults>oar env_reset ,env_keep += "PWD␣DISPLAY␣OAR_JOB_ID
␣OAR_JOB_KEY_FILE" , !requiretty ' >> /etc/sudoers
echo 'Defaults:oar env_reset ,env_keep += "DISPLAY␣SSH_CLIENT
␣SSH2CLIENT␣SHELL" , !requiretty ' >> /etc/sudoers

echo 'Cmnd_Alias OARCMD = /usr/local/oar/oarnodes ,
 /usr/local/oar/oarstat ,\ ' >> /etc/sudoers
echo '/usr/local/oar/oarsub , /usr/local/oar/oardel ,
 /usr/local/oar/oarhold ,\ ' >> /etc/sudoers
echo '/usr/local/oar/oarnotify , /usr/local/oar/oarresume ,
 /usr/local/oar/oarsh ' >> /etc/sudoers
echo '%oar ALL=(oar) NOPASSWD: OARCMD' >> /etc/sudoers

echo 'oar ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers

echo 'Defaults:www-data   env_keep += "SCRIPT_NAME␣SERVER_NAME
␣SERVER_ADMIN␣HTTP_CONNECTION␣REQUEST_METHOD␣CONTENT_LENGTH
␣SCRIPT_FILENAME␣SERVER_SOFTWARE␣HTTP_TE␣QUERY_STRING␣REMOTE_PORT
␣HTTP_USER_AGENT␣SERVER_PORT␣SERVER_SIGNATURE␣REMOTE_ADDR
␣CONTENT_TYPE␣SERVER_PROTOCOL␣PATH␣REQUEST_URI␣GATEWAY_INTERFACE
␣SERVER_ADDR␣DOCUMENT_ROOT␣HTTP_HOST" ' >> /etc/sudoers

echo 'www-data ALL=(oar) NOPASSWD: /usr/local/oar/oar-cgi ' >> /etc/sudoers

#Configuring ssh-server to accept oar connections without password

echo '#CONFIGURTION FOR OAR' >> /etc/ssh/sshd_config
echo 'AcceptEnv OAR_CPUSET SUDO_JOB_USER' >> /etc/ssh/sshd_config
echo 'PermitUserEnvironment yes' >> /etc/ssh/sshd_config
echo 'UseLogin no' >> /etc/ssh/sshd_config
echo 'AllowUsers oar' >> /etc/ssh/sshd_config
echo 'X11Forwarding yes' >> /etc/ssh/sshd_config
echo 'X11UseLocalhost no' >> /etc/ssh/sshd_config

mkdir /home/oar/.ssh
chown oar /home/oar/.ssh
chgrp oar /home/oar/.ssh
chmod 700 /home/oar/.ssh

#Creating and configuring ssh oar_key
su oar -c "echo␣-e␣'/home/oar/.ssh/id_rsa\n\n\n'␣|␣ssh-keygen"

echo 'Host *' > /home/oar/.ssh/config
echo '   ForwardX11 no' >> /home/oar/.ssh/config
echo ' StrictHostKeyChecking no' >> /home/oar/.ssh/config
echo '        PasswordAuthentication no' >> /home/oar/.ssh/config
echo '        AddressFamily inet ' >> /home/oar/.ssh/config

```

```

buf=`cat /home/oar/.ssh/id_rsa.pub`

echo "environment=\"OAR_KEY=1\" $buf" >/home/oar/.ssh/authorized_keys

##Giving appropriate permissions
chown oar:oar /home/oar/.ssh/id_rsa /home/oar/.ssh/id_rsa.pub
chown root:root /home/oar/.ssh/authorized_keys /home/oar/.ssh/config

chmod 644 /home/oar/.ssh/authorized_keys /home/oar/.ssh/config
 /home/oar/.ssh/id_rsa.pub
chmod 600 /home/oar/.ssh/id_rsa


#Install

cp /usr/local/uss-factory/conf/oar-2.2.13.tar.gz /tmp
cd /tmp ; tar -xvzf oar-2.2.13.tar.gz
cd /tmp/oar-2.2.13
mkdir -p /var/lib/oar
make server-install
make node-install
make user-install
cd / ; rm -Rf /tmp/oar-2.2.13

#Configuring Mysql data base

service sshd restart
echo -e "root\necureuil" | oar_mysql_db_init
oarnodesetting -a -h `hostname` -p cpuset=0
Almighty &
```