

Put some *Green* in your Grid'5000 experiments

https://www.grid5000.fr/mediawiki/index.php/Put_Some_Green_In_Your_Experiments

Introduction

This practical session is focused on energy aspects in Grid'5000. This session is based on the 150 energy sensors (wattmeters) infrastructure provided by Grid'5000/Aladdin and recently deployed on Grid'5000 Lyon site and some software framework provided by the GREEN-NET ARC project (<http://www.ens-lyon.fr/LIP/RESO/Projects/GREEN-NET/>).

The session will be divided into three main parts:

Energy usage exposing: the goal is to learn how to observe and collect energy information of basic Grid'5000 node operations (i.e. deployment of a node, intensive CPU usage, disk access and high performance communication).

Energy Efficient HPC: this aims to experiment and to see the energy aspects of some MPI programming design and choices.

Virtualization and Energy: the purpose is to analyze the energy impact of virtualization and of virtual machines mapping choices with a focus on live migration.

1 Energy Usage Exposing

1.1 Image Deployment

First, you need to log on Lyon site:

```
ssh login@access.lyon.grid5000.fr
```

Then, we will give you five nodes, the first three ones will be used with the first image and the two remaining ones will be used with the second image that we will use on section 3.

image	TPG5K-green-image1			TpGrid5000	
node					

To deploy TPG5K-green-image1 on your three nodes, you can create a file called *machines1* with the name of your nodes.

machines1 file

```
node1.lyon.grid5000.fr
node2.lyon.grid5000.fr
node3.lyon.grid5000.fr
```

It's time to look at the machine time (with the `date` command) and to write it there for example:


Deployment start time	
-----------------------	--

Then, you can launch the deployment by using the following command:

```
kadeploy3 -e TPG5K-green-image1 -u acorgerie -f machines1 -k
```

When the deployment is finished, you can log on the nodes:

```
ssh root@node
```

 **Comment:** Help for deployment can be found here:
https://www.grid5000.fr/mediawiki/index.php/Deploy_environment

1.2 Access to Energy Information

The webpage that gives all the energy information is there:

<https://helpdesk.grid5000.fr/supervision/lyon/wattmetre/>

(Grid'5000 account required). This first page explains how this energy tool works. It can be useful to read it!

If you click on the '**Live Monitoring**' tab, you will see one RRD¹ graph per node. You can click on the graph concerning one of your nodes, and you will access the different provided views: hour, day, week, month and year view. Those graphs are updated every minutes.

On the '**Logs Access**' tab, a "log on demand" service is provided: you need to fill the start and end time wanted, the nodes and the increment which represents the time period between two measures that you will obtain (in seconds). You will obtain a directory (with a permanent address) which will contain:

- a log file for each requested node (.dat extension),
- an svg graph for each requested node,
- a log file with the global consumption of each requested node during the requested time period (.log extension) and
- a tar.gz file containing all the previous files.

Now, it's time to test it (please, ask only for your node and for short periods of time; this infrastructure is still under development)! You can put the start time you have just written on the previous page, and you check the boxes corresponding to your three nodes.

✗ Question 1.1 (Graph observation). *Does the boot cause an energy consumption increase? If yes, can you see the different steps of the kadeploy deployment? If no, is this normal?*

1.3 Basic Operations

The goal here is to observe the energy consumption of basic operations of Grid'5000 applications. In order to do this, you can use the live monitoring tools or the logs on demand tools tested just before.

1.3.1 Idle Nodes

✗ Question 1.2. *What is the consumption of your nodes when they are idle? Is it high?*

We will call this value the *idle consumption* (ie. the consumption when the node is booted but idle). You can launch a `top` command to verify that your nodes are idle.

✗ Question 1.3. *Do your three nodes have the same idle consumption? How can you explain that?*

¹Round-Robin Database <http://oss.oetiker.ch/rrdtool/>

1.3.2 Disk Access Intensive Application

Go to directory `/home/user/basic/`. For each experiment you will launch in this section, a `logs` file will be created and will contain the start time of your experiment.

To perform a disk intensive application, you can make your own disk intensive application or launch the following command on the nodes:

```
./hdparm.sh 120 &
```

where '120' is the length of the experiment in seconds. This application uses a loop with the `hdparm`² command.

✗ Question 1.4. *Does this application increase the energy consumption of your node? What is the difference in percentage between the consumption of your nodes during this experiment and when they are idle? Is it significant?*

1.3.3 CPU Intensive Applications


CPU is considered as the most energy consuming component of computing nodes.

✗ Question 1.5. *How many CPUs do you have on your nodes?*

You can launch a cpu intensive application on the nodes with following command:

```
./cpuburn.sh 120 &
```

where '120' is the length of the experiment in seconds.

 This application uses `cpuburn` (`burnP6`) which is not so good for the nodes, so please do not over-use it!

✗ Question 1.6. *Does this application increase the energy consumption of your node? What is the difference in percentage between the consumption of your nodes during this experiment and when they are idle? Is it significant?*


✗ Question 1.7. *Is this experiment more or less energy consuming than the disk intensive one?*

Another cpu intensive application is provided. It can be launched on the nodes with the following command:

```
./stress.sh 120 &
```

where '120' is the length of the experiment in seconds.

✗ Question 1.8. *Do you obtain the same energy consumption for the two cpu intensive experiments? Why?*

 **Comment:** You can have a look on the cpu usage with the `htop` command while running the two experiments.

You can test with your own cpu intensive application.

²For more information: `man hdparm`

1.3.4 Network Intensive Application

To launch the network intensive application, you need two nodes: one client and one server. On the server, the command is:


```
./iperf-server-UDP.sh 120 &
```

where '120' is the length of the experiment in seconds. This experiment uses `iperf` with UDP traffic.

And then quickly, you launch on the client the following command:

```
./iperf-client-UDP.sh 120 serveraddress &
```

where '120' is the length of the experiment in seconds and `serveraddress` is either the IP address of the server node or its full name (`sagittaire-76.lyon.grid5000.fr` for example).

 **Comment:** You can fix the asked bandwidth in `iperf-client-UDP.sh`. It's the value after the `-b` option. By default, we have set it to 1GB/s.

✗ Question 1.9. *Does this application increase the energy consumption of your node? What is the difference in percentage between the consumption of your nodes during this experiment and when they are idle? Is it significant?*

✗ Question 1.10. *Is there a difference in terms of energy consumption between the client and the server (in percentage)?*

✗ Question 1.11. *Is this experiment more consuming than the previous ones?*

You can test the consumption with TCP traffic with the following command on the server:

```
./iperf-server-TCP.sh 120 &
```


where '120' is the length of the experiment in seconds.

And then quickly, you launch on the client the following command:

```
./iperf-client-TCP.sh 120 serveraddress &
```

where '120' is the length of the experiment in seconds and `serveraddress` is either the IP address of the server node or its full name (`sagittaire-76.lyon.grid5000.fr` for example).

✗ Question 1.12. *Is there a difference in terms of used bandwidth between these two experiments (with UDP and TCP traffic)?*

 **Comment:** You can see this information in the `logs` file (the output of the `iperf` commands are re-directed in this file).

✗ Question 1.13. *Is there a difference in terms of energy between these two experiments (with UDP and TCP traffic)? Where does it come from?*

1.4 First summary

✗ Question 1.14. *What is the most energy consuming application?*

✗ Question 1.15. *What does seem to be the highest value for the energy consumption of your nodes in Watts?*

2 Energy Efficient HPC

2.1 MPI Experiments

A programmer designs a MPI client-server application as presented in Figure 1. 3 processes run concurrently:

- 1 server is giving tasks to 2 clients,
- client 1 is computing small tasks (S Tasks),
- client 2 is computing XL Tasks (1 XL task = 2 S tasks).

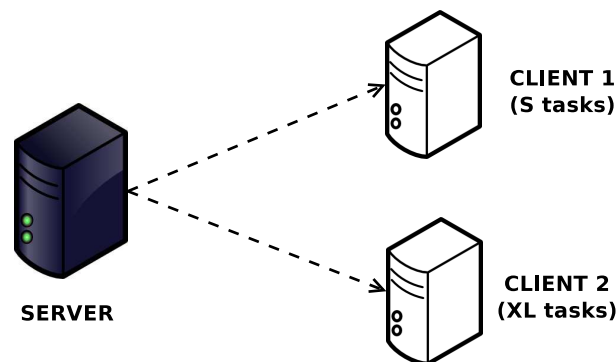


Figure 1: MPI experiment

2.2 MPI Scenario n°1: dumb application

Go to `/home/user/mpi/` directory.

The programmer is not very good and produces a first version of the MPI application: `dumb_version.c`. Have a look on the program in order to understand what the programmer has done.

Compile your dumb application with `mpicc` on each node:

```
mpicc -o version_dumb version_dumb.c
```

⚠ Do not forget to note the start time of your experiments!

Launch you MPI application locally (on the server for example):

```
mpirun -np 3 version_dumb
```

👉 **Comment:** This application is end-less, so you can kill it (with `Ctrl+d`) when you are tired.

✘ **Question 2.1.** *What is the energy profile that you observed?*

Copy the `machines1` file on the server in `/home/user/mpi/` and run your application on the 3 different machines:

```
mpirun -np 3 -machinefile machines1 version_dumb
```

✘ **Question 2.2.** *What is the energy profile that you observed?*

✘ **Question 2.3.** *Is this scenario energy-efficient? By the way, how would you define "energy efficiency"?*

2.3 MPI Scenario n°2: a greener MPI program

✘ **Question 2.4.** *How can you improve the proposed application in order to reduce idle time for clients?*

👉 **Comment:** Idea: improve the server reactivity!

OK, you don't know how to improve your MPI application, have a look on the `version_top.c` MPI program (in `/home/user/mpi/`).

As previously, you need to compile the top version on each node:

```
mpicc -o version_top version_top.c
```

Then, you can launch it on the three nodes:

```
mpirun -np 3 -machinefile machines1 version_top
```

✘ **Question 2.5.** *What is the impact of your MPI modifications for the server in terms of energy? Impact on clients' consumption?*

✘ **Question 2.6 (Bonus).** *How can you improve the energy efficiency of your own MPI application?*

3 Virtualization and Energy

For the experiments described in this section, you will need the two remaining nodes.

3.1 Image Deployment

The first step is to deploy the Grid'5000 environment that contains the Xen hypervisor (i.e. the disk image with Dom0). Assuming that you have a file named `machines2` with the list of nodes, type the following command:

```
kadeploy3 -e TpGrid5000 -f machines2 -u omonard -k
```

Second, you will have to customise the environment a little in both machines. Log into each machine:

```
ssh node
```

After that, logged as root, you will have to give yourself rights to execute the `xm` command as sudo. That is, you need to add "youruser ALL=NOPASSWD: /usr/sbin/xm" to the sudoers file (where *youruser* is your login). You also need to start the `xend` daemon:

```
su -
visudo
/etc/init.d/xend start
exit
```

At this stage, to run VMs you would need to have the disk images of the guest domains (i.e. domUs). To make your life easier, we have prepared a few guest images that you will deploy. You just need to copy them to a directory termed `tp` in your user area:

```
cd
mkdir tp
cp /home/lyon/mdiasdeassuncao/tp/* tp/
chmod 666 tp/*
```

You are now ready to run VMs on the nodes you reserved!

3.2 Virtual Machine Cost

We are going to start a VM and observe the energy consumed by the host where the VM executes. Before you run the VM, you might want to write down the time when you started:

VM deployment start time	
--------------------------	--

Assuming you are logged to one of the Xen nodes and in the **tp** directory where you have placed the disk images, to start the VM you should execute the following command:

```
cd tp/
sudo xm create debno01.cfg
```

You can check the Xen domains currently in execution by typing:

```
sudo xm list
```

There are a few things about which you should be aware when configuring the network interfaces of your VMs. As the purpose of this TP is not to explain in detail how to use Xen on Grid'5000, we will use the **console** to connect to our VMs. To connect to our VM, we use:

```
sudo xm console vml
```

You can login using the **demo** user and the default Grid'5000 password. We are now going to run **cpuburn**, described beforehand, for 20 seconds on this VM. Hence, once logged as **demo** on the VM, execute the following commands:

```
cd /home/demo/cpuburn
./cpuburn.sh 20
```

After that, you can log out of the VM (Ctrl+Alt+Gr+J) and shut it down:

```
sudo xm shutdown vml
```

✗ **Question 3.1.** *Can you observe the energy consumed by booting and shutting down a VM?*

✗ **Question 3.2.** *Is the energy consumption of using one CPU for the VM significant?*

3.3 Capping and Pinning on a Virtual Machine

We are now going to start VMs that run a CPU intensive application (i.e. `cpuburn`) once they are initialised. We are going to observe the effect of changing a few parameters of Xen hypervisor's scheduler, such as throttling the CPU usage. But before you start, do not forget to write down the deployment start time:


VM deployment start time	<input type="text"/>
--------------------------	----------------------

To start the first VM, logged in one of the Xen nodes type the following command:

```
sudo xm create deblo01.cfg
```

Now lets change the CPU utilisation cap for this VM. Wait around 10 seconds between each of the commands below:

```
sudo xm sched-credit -d vm1 -c 100
sudo xm sched-credit -d vm1 -c 70
sudo xm sched-credit -d vm1 -c 50
sudo xm sched-credit -d vm1 -c 20
sudo xm sched-credit -d vm1 -c 100
```

 **Comment:** The `-c` option sets the cap. For example, with `-c 50`, your VM will use 50% of the CPU allocated to it. For more information, see <http://wiki.xensource.com/xenwiki/CreditScheduler>.

✗ Question 3.3. *Is it possible to observe the energy consumed by running the VM with a CPU intensive application?*

✗ Question 3.4. *Is the impact of CPU throttling noticeable?*

Lets start another VM:

```
sudo xm create deblo02.cfg
```

Each VM is assigned one Virtual CPU (VCPU). What if we want to consolidate the workload to save energy and make the two VMs share the same physical CPU? Lets do that by **pinning** the VCPUs of both VMs to the same physical CPU:

```
sudo xm vcpu-pin vm1 0 0
sudo xm vcpu-pin vm2 0 0
```

 Do not shut down the VMs just yet! We will migrate one of them soon.

✗ Question 3.5. *Pinning the VMs to share the same CPU has any impact on the energy consumed by the physical host?*

3.4 Live Migration

So far you have used only one of the nodes to run your VMs. Imagine that for some reason, this host becomes overloaded and you need to migrate one of the VMs to another node. We are now going to do that and observe the energy consumed during the migration. Again, we do not mean to be annoying, but it would be nice to write down the time when you start the migration:

VM migration start time	
-------------------------	--

To migrate **vm1** to **second_host**, use the following command:

```
sudo xm migrate -l vm1 second_host
```

You can now shut the VMs down if you wish.

✘ Question 3.6. *Can you observe the energy consumed when migrating a VM?*

Conclusion

Thank you for your participation. Any comments and feedbacks are welcome!