Using Computational Resources in Optimization and Statistics

Sébastien Martin

MIT

Tuesday 01/24/2017

Heavy Computations

In Optimization and Statistics, we often need a lot of computational power:

- Machine Learning on large datasets
- Hard optimization problems, mixed integer programming

Heavy Computations

In Optimization and Statistics, we often need a lot of computational power:

- Machine Learning on large datasets
- Hard optimization problems, mixed integer programming

Or repetitive computations:

- Parameter tuning
- Benchmarking

Limitations of a Personal Computer

Using your personal computer may seem simple, but there are serious limitations:

- Limited memory (Big Data, large matrices...)

Limitations of a Personal Computer

Using your personal computer may seem simple, but there are serious limitations:

- Limited memory (Big Data, large matrices...)
- Limited computational power.

Limitations of a Personal Computer

Using your personal computer may seem simple, but there are serious limitations:

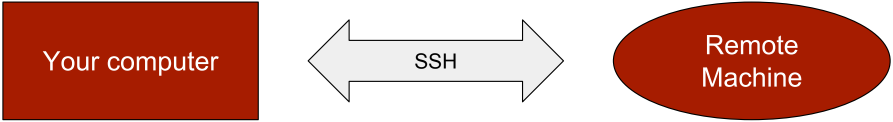
- Limited memory (Big Data, large matrices...)
- Limited computational power.
- Limited number of machines/cores.

Limitations of a Personal Computer

Using your personal computer may seem simple, but there are serious limitations:

- Limited memory (Big Data, large matrices...)
- Limited computational power.
- Limited number of machines/cores.
- Limited time available. (you want to use your laptop for other things too..)

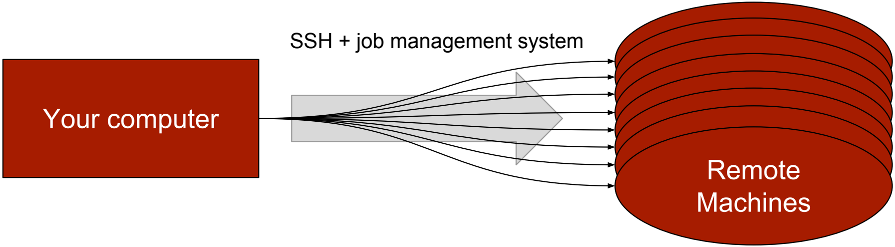
How does it work : using a remote computer



An interactive remote control is the easiest way to use another computer.

- We use SSH (see lecture 1) to control a terminal on a remote machine through our computer.
- We can use the console to do almost anything on the remote computer: create files, run a program, use Julia...
- It is also possible to use graphical softwares like R-Studio or Matlab.
- It works with any computer, including your own machines.

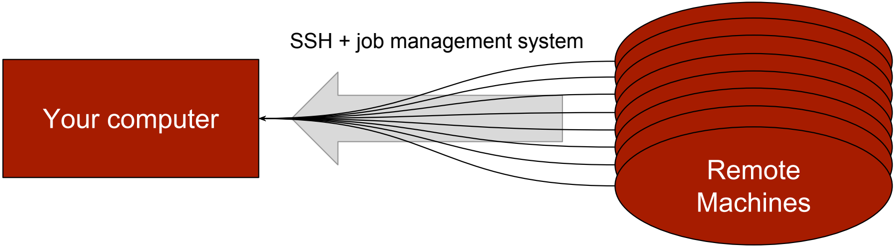
How does it work : using a cluster



When a computing cluster is available, we can run multiple “jobs” thanks to a job management system.

- Allows us to use multiple remote machines, manages demand and resources available.
- Different job management systems exist, I will use the example of Slurm, available to Sloan and ORC students on the cluster Engaging.
- More complex to use, but far more powerful.

How does it work : using a cluster



When a computing cluster is available, we can run multiple "jobs" thanks to a job management system.

- Allows us to use multiple remote machines, manages demand and resources available.
- Different job management systems exist, I will use the example of Slurm, available to Sloan and ORC students on the cluster Engaging.
- More complex to use, but far more powerful.

Resources

There are several ways to have access to a remote computer or a cluster.

- Another personal computer you own.
- Athena at MIT.
- Resources of your department/lab (Engaging at Sloan).
- Paying options: cloud computing (Amazon AWS, Google Cloud Computing...).

Why Should I Use this

For research:

- Tackle bigger problems in Stats and Optimization.
- Parallelize your computations (parameter grid search in ML for example)!
- Longer computational times: run overnight or for a whole week!
- Bigger Datasets become manageable.
- Can be very simple to use with interactive sessions (RStudio...)

In general:

- Useful skill, at the age of cloud computing.
- Good practice to learn how to use console (and GitHub).

What Machine? Engaging

Engaging is a powerful computing cluster for MIT Sloan affiliates, request an account at `stshelp@mit.edu`. It uses the job management system Slurm, that we will use in this presentation.

Connecting to Engaging

Hostname	`eosloan1.mit.edu`
Username	MIT Kerberos username (not your email address)
Password	MIT Kerberos password

`wikis.mit.edu/confluence/display/sloanrc/Engaging+Platform`

What Machine? Athena

Any MIT affiliate can access the Athena distributed computing environment, you can use it to follow if you do not have access to Engaging.

Connecting to Athena

Hostname	`athena.dialup.mit.edu`
Username	MIT Kerberos username (not your email address)
Password	MIT Kerberos password

`http://web.mit.edu/dialup/www/ssh.html`

SSH

As seen in Lecture 1, you can connect to the remote machine through SSH:

```
[sebastien@seblaptop ~]$ ssh semartin@eosloan1.mit.edu
Password:
[semartin@eosloan1 ~]$
```

SSH

As seen in Lecture 1, you can connect to the remote machine through SSH:

```
[sebastien@seblaptop ~]$ ssh semartin@athena.dialup.mit.edu
Password:
semartin@buzzword-bingo:~$
```

SSH

As seen in Lecture 1, you can connect to the remote machine through SSH:

```
[sebastien@seblaptop ~]$ ssh semartin@athena.dialup.mit.edu
Password:
semartin@buzzword-bingo:~$
```

Tip

If you connect regularly to the same remote, you can set up a password-less login (see Google). You can also set-up an alias instead of typing the full address each time.

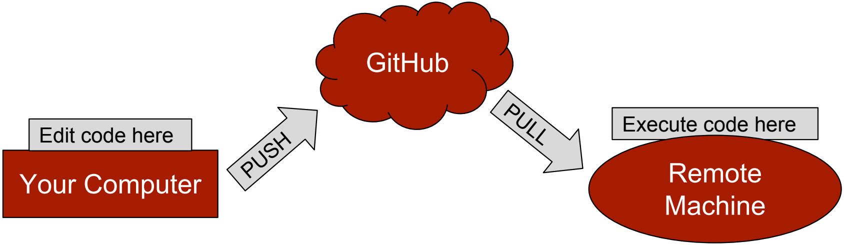
Using the Terminal

Once you are connected to the remote machine, you can use the terminal to interact with it:

```
[semartin@eosloan1 ~]$ pwd
/home/semartin
[semartin@eosloan1 ~]$ mkdir testfolder
[semartin@eosloan1 ~]$ cd testfolder
[semartin@eosloan1 testfolder]$ pwd
/home/semartin/testfolder
```

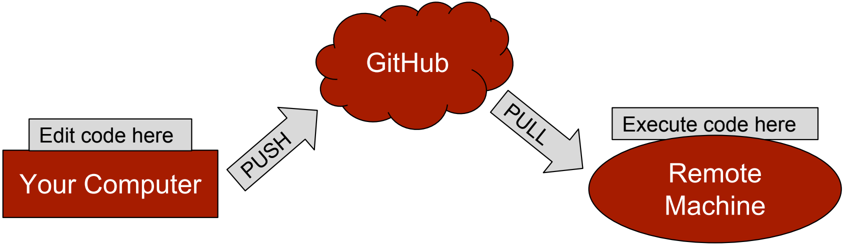
Using Git/GitHub to Synchronize Code

If you need to run code on the remote machine, an elegant way is to use git and GitHub:



Using Git/GitHub to Synchronize Code

If you need to run code on the remote machine, an elegant way is to use git and GitHub:



```
[semartin@eosloan1 testfolder]$ git clone \
> https://github.com/sebmart/IAP2017-computing-ressources.git
Cloning into 'IAP2017-computing-ressources'...
```

Using Screen to Keep your Session

By default, your session is lost when your ssh connection ends. Linux Screen can be used to close the connection while letting things run on the remote, and restart where you were.

Using Screen to Keep your Session

By default, your session is lost when your ssh connection ends. Linux Screen can be used to close the connection while letting things run on the remote, and restart where you were.

Start screen session: similar to starting a new terminal

```
[semartin@eosloan1 testfolder]$ screen
```

Detach the screen session before closing SSH: `Ctrl+A then D`: goes back to previous terminal. Then close the SSH connection (`Ctrl+D`)

Restart the SSH connection and re-attach your screen session as it was before:

```
[semartin@eosloan1 testfolder]$ screen -r
```

Useful commands

- Downloading from the Internet (useful to install software, or download datasets): wget

  ```
  $ wget http://data.com/awesomedata.csv
  ```

- Transferring files between the local and remote machines: Secure Copy scp

  ```
  $ scp mylocalfile.txt semartin@eosloan1.mit.edu:~
  ```

Useful commands

- Downloading from the Internet (useful to install software, or download datasets): wget

```
$ wget http://data.com/awesomedata.csv
```

- Transferring files between the local and remote machines: Secure Copy scp

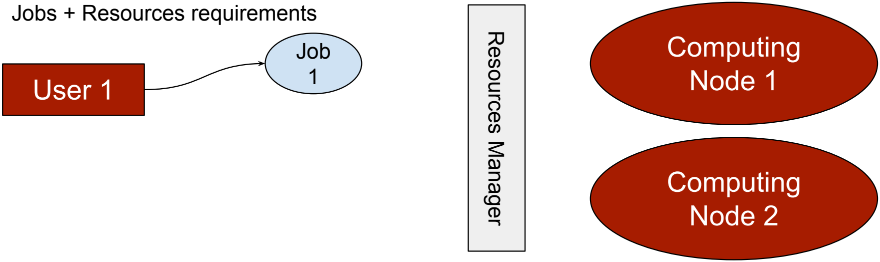
```
$ scp mylocalfile.txt semartin@eosloan1.mit.edu:~
```

For advanced users

Use a search engine to learn more about this !

- screen has lots of other advanced features (multiple windows...).
- tmux (terminal multiplexer) is like screen with lots of additional functionalities (panes...)
- sshfs allows you to access the files of the remote machine as if they were on your own (quite magic)!

Running Jobs on a Computing Cluster

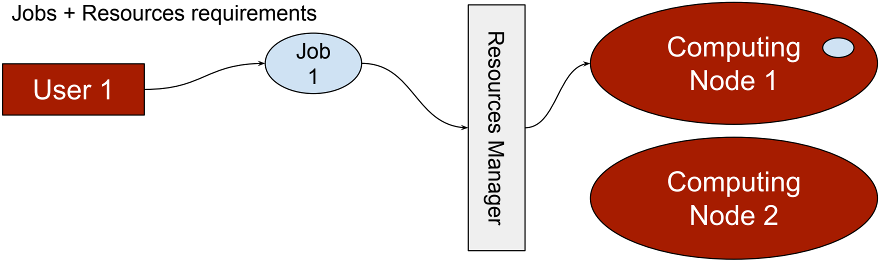


Job A program to be executed. (running a R/Julia/Python source file, a bash script, etc..)

Resources Resources required to run the job (memory, cpus, time, softwares...)

The Job/Resource manager allocates the available resources of the computing nodes of the cluster to the different users.

Running Jobs on a Computing Cluster

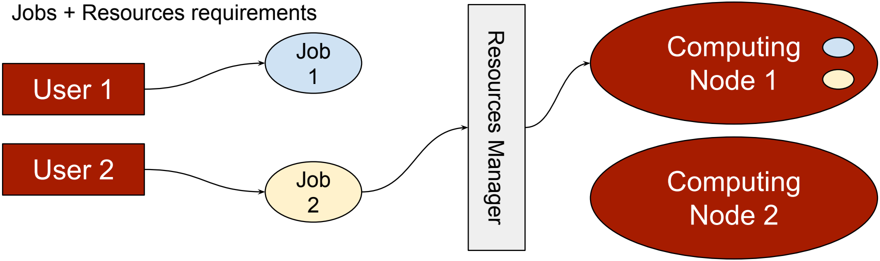


Job A program to be executed. (running a R/Julia/Python source file, a bash script, etc..)

Resources Resources required to run the job (memory, cpus, time, softwares...)

The Job/Resource manager allocates the available resources of the computing nodes of the cluster to the different users.

Running Jobs on a Computing Cluster

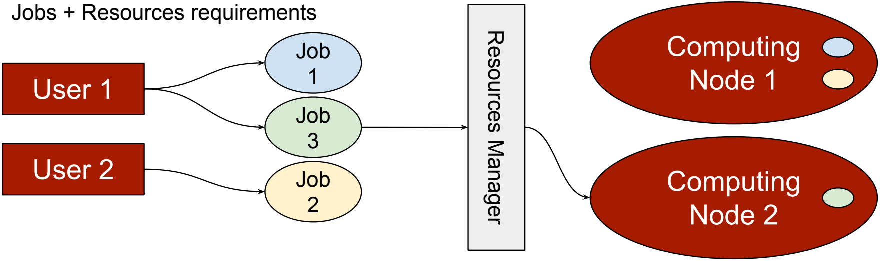


Job A program to be executed. (running a R/Julia/Python source file, a bash script, etc..)

Resources Resources required to run the job (memory, cpus, time, softwares...)

The Job/Resource manager allocates the available resources of the computing nodes of the cluster to the different users.

Running Jobs on a Computing Cluster



- Job: A program to be executed. (running a R/Julia/Python source file, a bash script, etc..)
- Resources: Resources required to run the job (memory, cpus, time, softwares...)

The Job/Resource manager allocates the available resources of the computing nodes of the cluster to the different users.

Module Loading

Engaging uses the resource manager SLURM. Lots of software are installed on the cluster. To use them, we use the special commands:

- Show all modules (software)

```
[semartin@eosloan1]$ module avail
```

- Search for modules

```
[semartin@eosloan1]$ eo-module-find julia
```

- Load a module

```
[semartin@eosloan1]$ module load engaging/julia/0.5.0
```

Running a Job on Engaging - srun

To ask SLURM, we use the command `srun`.

```
$ srun --partition=<mypartition> <parameters> <job>
$ srun --partition=sched_mit_sloan --mem=4G \
> julia myjuliafile.jl
```

- The main resource parameters are:

Memory	`--mem=8G`
CPUs	`--cpus-per-task=2`
Time	`--time=3-12:00` for 3 days and 12 hours

- The partition is the name of the cluster you want to use. Sloan students have access to `sched_mit_sloan` for jobs lasting up to two weeks.

Monitoring Jobs

- You can check if your job is running or in the queue:

```
[semartin@eosloan1 ~]$ eo-show-myjobs
[semartin@eosloan1 ~]$ eo-show-alljobs
```

- Cancel a job:

```
[semartin@eosloan1 ~]$ scancel <job_number>
```

Running a Job on Engaging - sbatch

- A nice way to run jobs is to save them and all their parameter in a special bash file, and us sbatch to run them. To do so, create a file myjobname.sh containing:

```
#!/bin/bash
#SBATCH --cpus-per-task=2
#SBATCH --mem=4G
#SBATCH --partition=sched_mit_sloan
#SBATCH --time=3-12:00
module load engaging/julia/0.5.0
srun julia my_julia_file.julia
```

- The corresponding job can be run using "sbatch myjobname.sh"
- The output of your jobs (what would normally appear in the console) will be saved in a slurm-<yourjob>.out log file where you ran sbatch.

Multiple Jobs with sbatch

`sbatch` also allow to run several jobs simultaneously! In an “array” of jobs, each sub-job is associated with a unique number: from 1 to the total number of sub-jobs. This number can be used to run a different job, using the environment variable `SLURM_ARRAY_TASK_ID`. Here is an example with an array of 20 sub-jobs, where the Julia program is given the job number as a parameter.

```bash
#!/bin/bash
#SBATCH -a 1-20
#SBATCH --cpus-per-task=2
#SBATCH --mem=4G
#SBATCH --partition=sched_mit_sloan
#SBATCH --time=3-12:00
module load engaging/julia/0.5.0
srun julia my_julia_file.julia $SLURM_ARRAY_TASK_ID
```

Running Multiple Jobs Needs Organization

One way to manage results from multiple jobs:

- Do not be afraid of submitting too many jobs: SLURM will schedule them for you over time as resources become available.
- Do not over-estimate the resources that you need: your jobs will be scheduled faster if your requirements are low.
- Try to de-bug your program before running them on the cluster: it is far easier.
- Make each job write its output to a file, and once all jobs have finished, copy the output files on your computer and create a small program that aggregate them to your needs.

Other Things to Explore

There are a lot more possibilities with Engaging and other systems! Read the Engaging and SLURM documentation, use Google, etc.. For example:

- Running interactive jobs (ie with a console that you control)
- Running graphical interfaces (RStudio, SAS, Matlab...)
- Receive an email when your job completes
- Use Jupyter Notebooks on the cluster...