Introduction to Grid’5000

Chuyuan Li
10/11/2020

What is Grid’5000 and why would you use it?

This is a large-scale and flexible testbed for experiment-driven research.

We mainly interested in its large amount of resources:

- when you want to run a GPU-required machine learning task but you don't have GPU in your own computer
- when you run a time-consuming calculation and wish not to occupy 90% of your CPU all the time, *etc*.

For detailed description, refer to this link

Outline

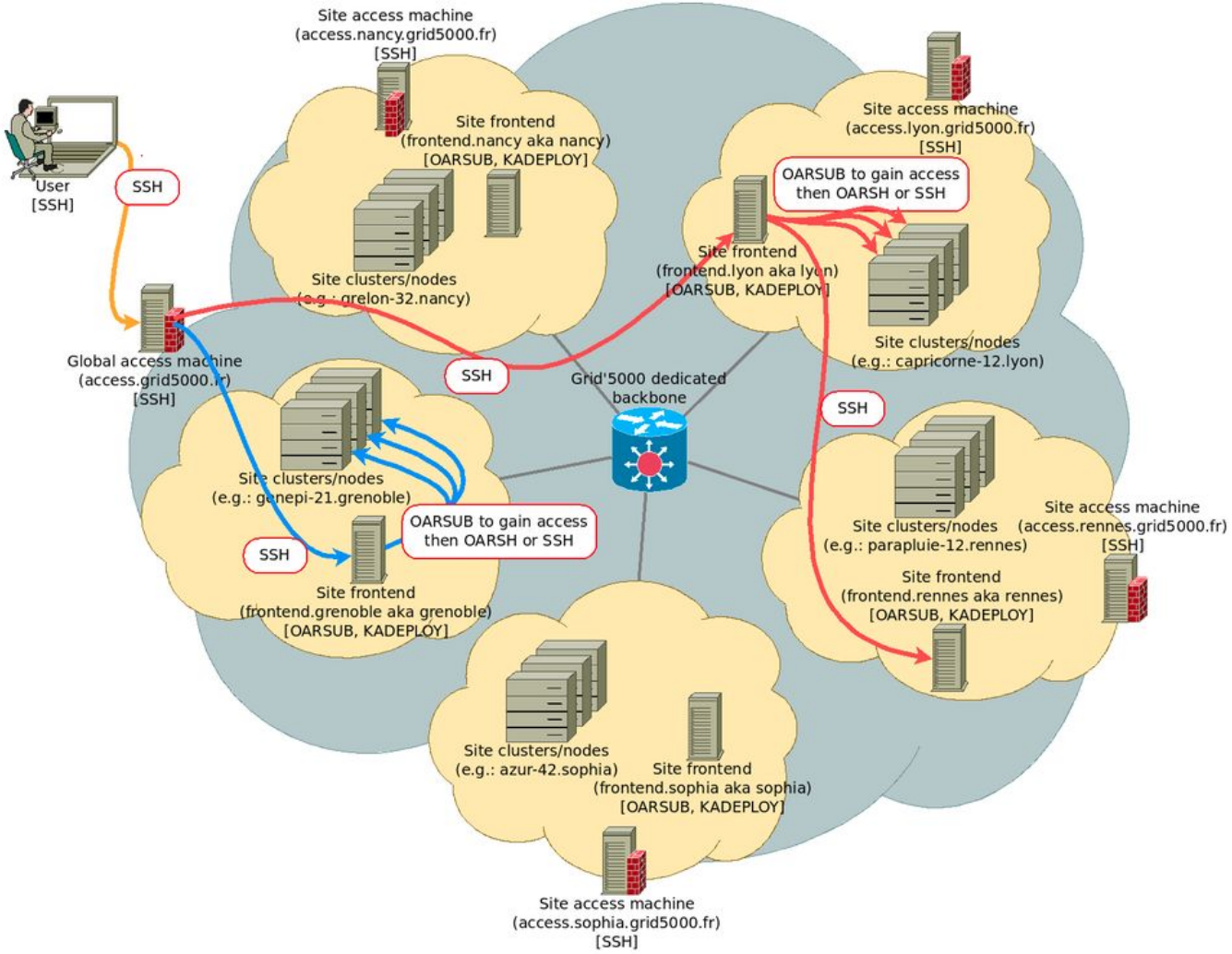
- Get an account of Grid’5000
- Connection with SSH key
- Basic concepts (cluster, node, host, core...)
- File/folder transfer
- Resources visualisation
- Resources reservation and management with OAR
- *TBD*

Before we start...

- Please check you have an account and can access to frontend
  - Open a terminal
  - type: ssh login@access.grid5000.fr
- THE site you will frequent:
  - https://www.grid5000.fr/w/Getting_Started

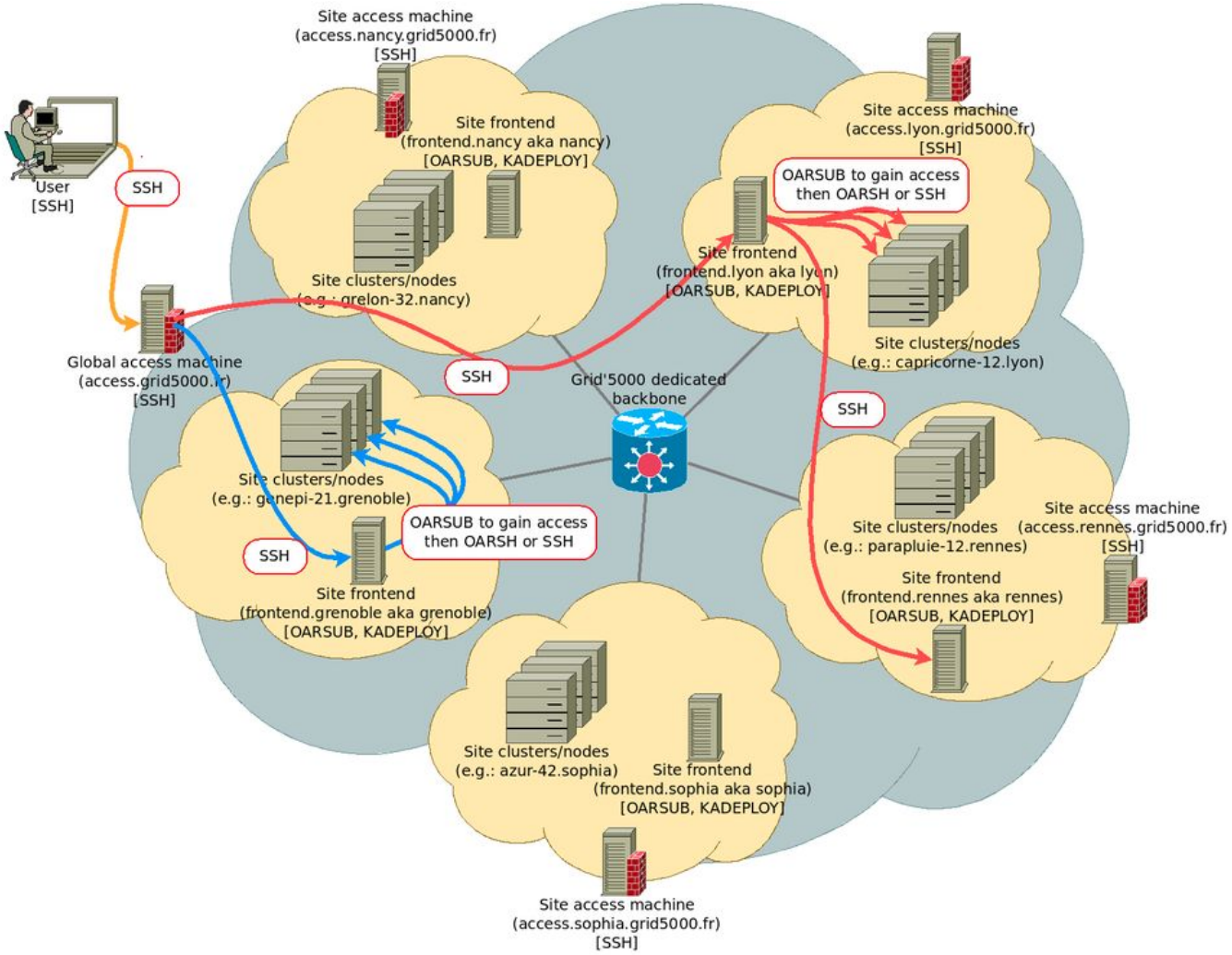
Big picture

Concept map


Site access machine
(access.nancy.grid5000.fr)
[SSH]
Site frontend
(frontend.nancy aka nancy)
[OARSUB, KADEPLOY]
Site clusters/nodes
(e.g.: grelon-32.nancy)
User
[SSH]
SSH
Global access machine
(access.grid5000.fr)
[SSH]
SSH
Grid'5000 dedicated
backbone
Site access machine
(access.lyon.grid5000.fr)
[SSH]
OARSUB to gain access
then OARSH or SSH
Site frontend
(frontend.lyon aka lyon)
[OARSUB, KADEPLOY]
Site clusters/nodes
(e.g.: capricorne-12.lyon)
SSH
Site clusters/nodes
(e.g.: genepi-21.grenoble)
OARSUB to gain access
then OARSH or SSH
SSH
Site frontend
(frontend.grenoble aka grenoble)
[OARSUB, KADEPLOY]
Site clusters/nodes
(e.g.: parapluie-12.rennes)
Site access machine
(access.rennes.grid5000.fr)
[SSH]
Site frontend
(frontend.rennes aka rennes)
[OARSUB, KADEPLOY]
Site clusters/nodes
(e.g.: azur-42.sophia)
Site frontend
(frontend.sophia aka sophia)
[OARSUB, KADEPLOY]
Site access machine
(access.sophia.grid5000.fr)
[SSH]

Concept map

- cluster
- nodes
- host
- core



Hardware in Nancy site

13 clusters, 374 nodes, 7784 cores, 323.3 TFLOPS

Cluster	Access Condition	Date of arrival	Nodes	CPU	Cores	Memory	Storage	Network	Accelerators
graffiti	**production** queue	2019-06-07	13	2 x Intel Xeon Silver 4110	8 cores/CPU	128 GiB	**479 GB SSD**	10 Gbps	4 x Nvidia RTX 2080 Ti
graoully	**production** queue	2016-01-04	16	2 x Intel Xeon E5-2630 v3	8 cores/CPU	128 GiB	**1 x 600 GB HDD** + 1 x 600 GB HDD	10 Gbps + 56 Gbps InfiniBand	
graphique	**production** queue	2015-05-12	6	2 x Intel Xeon E5-2620 v3	6 cores/CPU	64 GiB	**299 GB HDD**	10 Gbps + 56 Gbps InfiniBand	1: 2 x Nvidia Titan Black [2-6]: 2 x Nvidia GTX 980
graphite		2013-12-05	4	2 x Intel Xeon E5-2650	8 cores/CPU	256 GiB	**1 x 300 GB SSD** + 1 x 300 GB SSD	10 Gbps + 56 Gbps InfiniBand	Intel Xeon Phi 7120P
grappe	**production** queue	2020-08-20	16	2 x Intel Xeon Gold 5218R	20 cores/CPU	96 GiB	**480 GB SSD** + 8.0 TB HDD*	25 Gbps	
grcinq	**production** queue	2013-04-09	47	2 x Intel Xeon E5-2650	8 cores/CPU	64 GiB	**1.0 TB HDD**	1 Gbps + 56 Gbps InfiniBand	
grele	**production** queue	2017-06-26	14	2 x Intel Xeon E5-2650 v4	12 cores/CPU	128 GiB	**1 x 299 GB HDD** + 1 x 299 GB HDD	10 Gbps + 100 Gbps Omni-Path	2 x Nvidia GTX 1080 Ti
grimani	**production** queue	2016-08-30	6	2 x Intel Xeon E5-2603 v3	6 cores/CPU	64 GiB	**1.0 TB HDD**	10 Gbps + 100 Gbps Omni-Path	2 x Nvidia Tesla K40M
grimoire		2016-01-22	8	2 x Intel Xeon E5-2630 v3	8 cores/CPU	128 GiB	**600 GB HDD** + 4 x 600 GB HDD* + 200 GB SSD*	4 x 10 Gbps + 56 Gbps InfiniBand	
grisou		2016-01-04	51	2 x Intel Xeon E5-2630 v3	8 cores/CPU	128 GiB	**1 x 600 GB HDD** + 1 x 600 GB HDD	[1-48]: 1 Gbps + 4 x 10 Gbps 49: 4 x 10 Gbps [50-51]: 4 x 10 Gbps + 56 Gbps InfiniBand	
gros		2019-09-04	124	Intel Xeon Gold 5220	18 cores/CPU	96 GiB	**480 GB SSD** + 960 GB SSD*	2 x 25 Gbps	
grue	**production** queue	2019-11-25	5	2 x AMD EPYC 7351	16 cores/CPU	128 GiB	**479 GB SSD**	10 Gbps	4 x Nvidia Tesla T4
grvingt	**production** queue	2018-04-11	64	2 x Intel Xeon Gold 6130	16 cores/CPU	192 GiB	**1.0 TB HDD**	10 Gbps + 100 Gbps Omni-Path	

Link: https://www.grid5000.fr/w/Nancy:Hardware

Queues and Usage Policy

- *Default* queue
  - Daytime is dedicated to smaller-scale experiments
  - Large-scale jobs must be executed during nights or weekends
  - generally, using advance reservations
  - **Read carefully the rules** in case of violation of usage
- Production queue
  - Smaller set of resources
  - Only in Nancy site
  - More suited to long-running, non-interactive jobs
- More information, ref to UsagePolicy

Queues and Usage Policy

- discover daily allowance with:

  `usagepolicycheck -l [--sites site1,sites2]`

- check the jobs that have been counted using:

  `usagepolicycheck -v --start '2020-10-20 11:00:24 +0200' --end '2020-11-03 10:00:24 +0100'`

First connection

Connecting and moving around

- Basic steps to get in a site:
  - open a terminal
  - connect to access machine: `outside: ssh login@access.grid5000.fr`
  - specify a site: `access: ssh site`
  - put in your password
  - then we can view machine list in this site

Connecting and moving around

- Basic steps to get in a site:
  - connect to access machine: `outside: ssh login@access.grid5000.fr`

```
[chuyli@lisa27:~$ ssh cli@access.grid5000.fr
Linux access-north 4.9.0-12-amd64 #1 SMP Debian 4.9.210-1+deb9u1 (2020-06-07) x86_64
----- Grid'5000 - access-north.grid5000.fr -----

Welcome to Grid'5000

** Connect to a site:
$ ssh {grenoble,luxembourg,lyon,nancy,nantes,rennes,sophia}

** Useful links:
 - account management (password change): https://api.grid5000.fr/ui/account
 - homepage: https://www.grid5000.fr/mediawiki/index.php/Category:Portal:User
 - charter : https://www.grid5000.fr/mediawiki/index.php/Grid5000:UserCharter
 - support : https://www.grid5000.fr/mediawiki/index.php/Support

** Data on access.grid5000.fr :
 - your home directory on access machines (access-north and access-south)
   is not synchronized and should not be use to store data.
 - please use ssh forwarding to send data directly to sites or
 - (outside) $ scp files login@access.grid5000.fr:reims/ using the nfs mount point in your home
```

  - specify a site: `access: ssh nancy`

Connecting and moving around

- Basic steps to get in a site:
  - specify a site: `access: ssh nancy`

```
cli@access-north:~$ ssh nancy
cli@nancy's password:
Linux fnancy 4.19.0-12-amd64 #1 SMP Debian 4.19.152-1 (2020-10-18) x86_64
----- Grid'5000 - Nancy - fnancy.nancy.grid5000.fr -----

This site has 13 clusters (see: https://www.grid5000.fr/w/Nancy:Hardware)

Available in queue default
 - graphite  (2013): 4 nodes (2 CPUs Intel Xeon E5-2650, 8 cores/CPU, 256GB RAM, 2x279GB SSD, 1 x 10Gb Ethernet, 1 x 56Gb InfiniBand)
 - grimoire  (2016): 8 nodes (2 CPUs Intel Xeon E5-2630 v3, 8 cores/CPU, 128GB RAM, 5x558GB HDD, 186GB SSD, 4 x 10Gb Ethernet, 1 x 56Gb InfiniBand)
 - grisou    (2016): 51 nodes (2 CPUs Intel Xeon E5-2630 v3, 8 cores/CPU, 128GB RAM, 2x558GB HDD, 4 x 10Gb Ethernet, 1 x 1Gb Ethernet)
 - gros      (2019): 124 nodes (1 CPU Intel Xeon Gold 5220, 18 cores/CPU, 96GB RAM, 447GB SSD, 894GB SSD, 2 x 25Gb Ethernet)

Available in queue production
 - grcinq    (2013): 47 nodes (2 CPUs Intel Xeon E5-2650, 8 cores/CPU, 64GB RAM, 931GB HDD, 1 x 1Gb Ethernet, 1 x 56Gb InfiniBand)
 - graphique (2015): 6 nodes (2 CPUs Intel Xeon E5-2620 v3, 6 cores/CPU, 64GB RAM, 278GB HDD, 1 x 10Gb Ethernet, 1 x 56Gb InfiniBand)
 - graoully  (2016): 16 nodes (2 CPUs Intel Xeon E5-2630 v3, 8 cores/CPU, 128GB RAM, 2x558GB HDD, 1 x 10Gb Ethernet, 1 x 56Gb InfiniBand)
 - grimani   (2016): 6 nodes (2 CPUs Intel Xeon E5-2603 v3, 6 cores/CPU, 64GB RAM, 931GB HDD, 1 x 10Gb Ethernet, 1 x 100Gb Omni-Path)
 - grele     (2017): 14 nodes (2 CPUs Intel Xeon E5-2650 v4, 12 cores/CPU, 128GB RAM, 2x278GB HDD, 1 x 10Gb Ethernet, 1 x 100Gb Omni-Path)
 - grvingt   (2018): 64 nodes (2 CPUs Intel Xeon Gold 6130, 16 cores/CPU, 192GB RAM, 931GB HDD, 1 x 10Gb Ethernet, 1 x 100Gb Omni-Path)
 - graffiti  (2019): 13 nodes (2 CPUs Intel Xeon Silver 4110, 8 cores/CPU, 128GB RAM, 446GB SSD, 1 x 10Gb Ethernet)
 - grue      (2019): 5 nodes (2 CPUs AMD EPYC 7351, 16 cores/CPU, 128GB RAM, 446GB SSD, 1 x 10Gb Ethernet)
 - grappe    (2020): 16 nodes (2 CPUs Intel Xeon Gold 5218R, 20 cores/CPU, 96GB RAM, 447GB SSD, 7452GB HDD, 1 x 25Gb Ethernet)

** Useful links:
 - account management (password change): https://api.grid5000.fr/ui/account
 - homepage: https://www.grid5000.fr/mediawiki/index.php/Category:Portal:User
 - charter : https://www.grid5000.fr/mediawiki/index.php/Grid5000:UserCharter
 - support : https://www.grid5000.fr/mediawiki/index.php/Support

** Others sites:
$ ssh {grenoble,lille,luxembourg,lyon,nantes,rennes,sophia}
Last login: Mon Nov  9 20:29:44 2020 from 192.168.66.33
```

Tip: use SSH ProxyCommand

- In ~/.ssh/config:

```
Host g5k
  User USERNAME
  Hostname access.grid5000.fr
  ForwardAgent no
Host *.g5k
  User USERNAME
  ProxyCommand ssh g5k -W "$(basename %h .g5k):%p"
  ForwardAgent no
```

- Connect to any Grid5k node in one command
  - $ ssh nancy.g5k
  - $ ssh lyon.g5k

Transferring files to/from Grid’5000

- no BACKUP in g5k, so make sure your important files are stored somewhere outside
- In each site, by default 25 GiB storage
  - If needed, can demand for more space
  - *manage account -> homedir quotas -> request quota extension*
- ProxyCommand works with everything SSH-based
  - scp, sftp, rsync
- Prefer **rsync** than scp
  - Pipelined file transfers
  - More efficient on networks with large BDP (bandwidth * latency)

Transferring files to/from Grid'5000

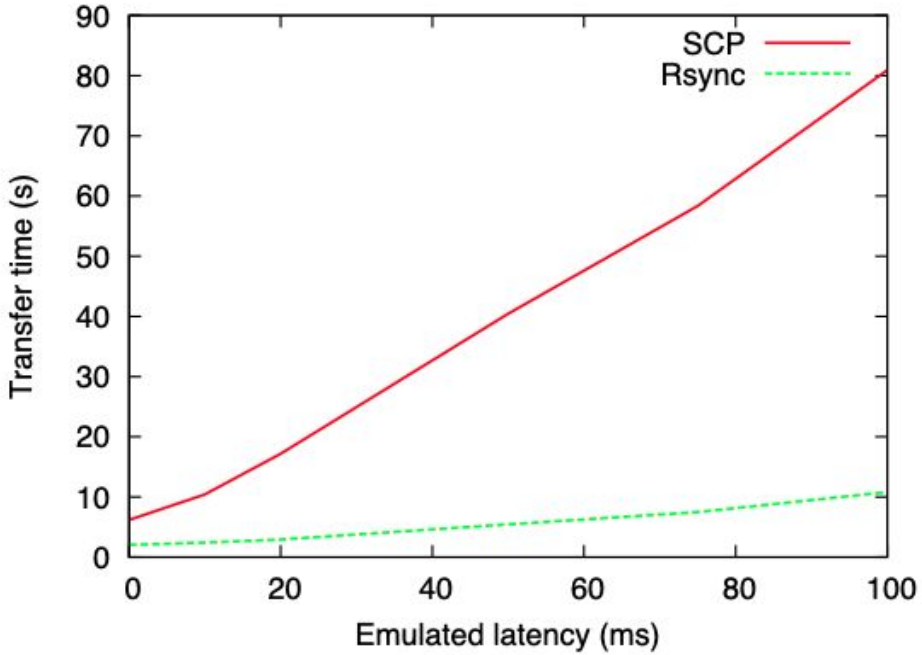
- scp
  - Copy file from local to remote:
    - `scp local_file remote_username@remote_ip:remote_file`
  - Copy folder from local to remote:
    - `scp -r local_folder remote_username@remote_ip:remote_folder`
  - Copy file from local to remote:
    - `scp remote_username@remote_ip:remote_file local_file`
  - Copy folder from remote to local:
    - `scp -r remote_username@remote_ip:remote_folder local_folder`
- Example
  - `local: $ scp -r /Users/chuyli/g5k_tuto/ cli@nancy.g5k:/home/cli/`
  - `local: $ scp cli@nancy.g5k:/home/cli/g5ktuto/show1.sh /Users/chuyli/g5k_tuto/`

Transferring files to/from Grid’5000

- rsync
  - Copy folder from local to remote:
    - `rsync -avzP local_folder remote_username@remote_ip:remote_folder`
  - Example:
    - `local: $ rsync -avzP /Users/chuyli/g5k_tuto cli@nancy.g5k:/home/cli/`
    - `local: $ rsync -avzP /Users/chuyli/g5k_tuto/ cli@nancy.g5k:/home/cli/`
    - Mind the difference between *local_folder* and *local_folder/*

  - Much faster than scp for large files, recommend for folder transfer
  - Syntaxe more complicated
  - To know more, check official link rsync

Transfer of 120 files (total : 2.1 MB) with SCP and Rsync
Bandwidth and Latency controlled using network emulator



Visualisation & Reservation

Visualizing Grid’5000 resources

- Several ways to learn about resources and their status
  - Monika: reservation state
  - Gantt: reservation history and forecast, very useful
  - Ganglia: resources usage (load, memory, CPU, network usage in last hour)
  - Platform events: show maintenance news
  - More info: ref nancy home site

Monika

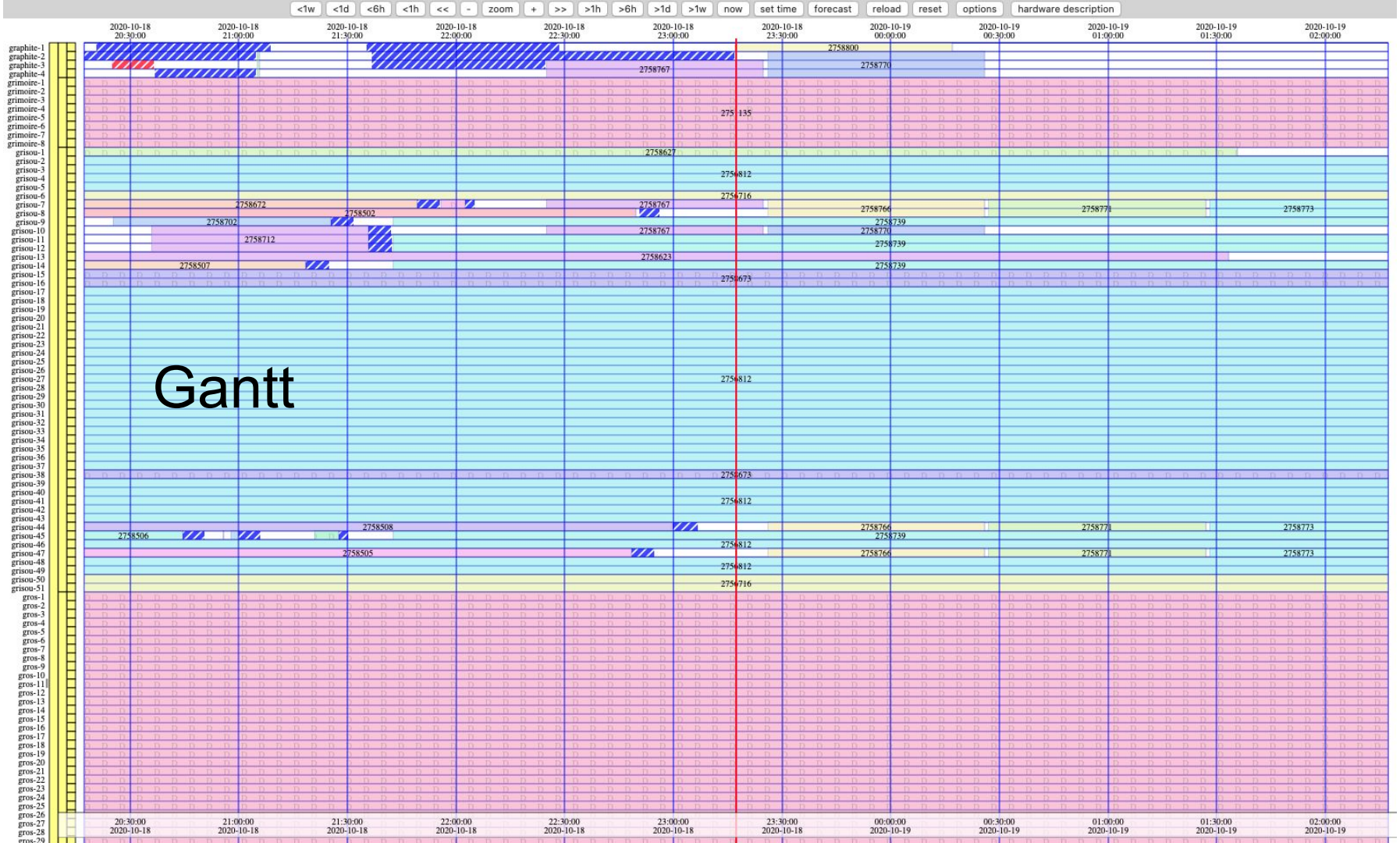
Grid5000 Nancy nodes

default summary

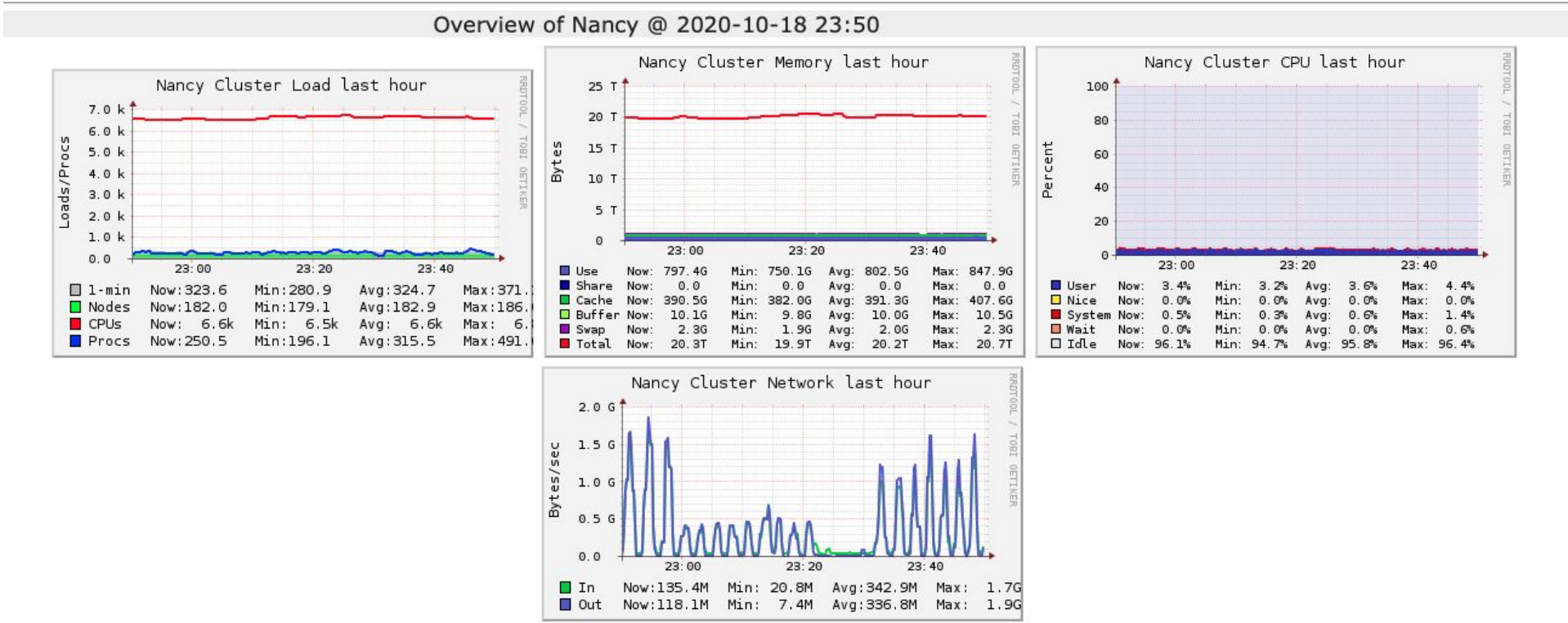
	Free	Busy	Total
network_address	3	183	187
resource_id	48	3176	3240

Reservations:

graphite-1	2758800	2758800	2758800	2758800	2758800	2758800	2758800	2758800	2758800	2758800	2758800	2758800	2758800	2758800	2758800
graphite-2	2758770	2758770	2758770	2758770	2758770	2758770	2758770	2758770	2758770	2758770	2758770	2758770	2758770	2758770	2758770
graphite-3	Free	Free	Free	Free	Free	Free	Free	Free	Free	Free	Free	Free	Free	Free	Free
graphite-4	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent
grimoire-1	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135
grimoire-2	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135
grimoire-3	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135	2751135

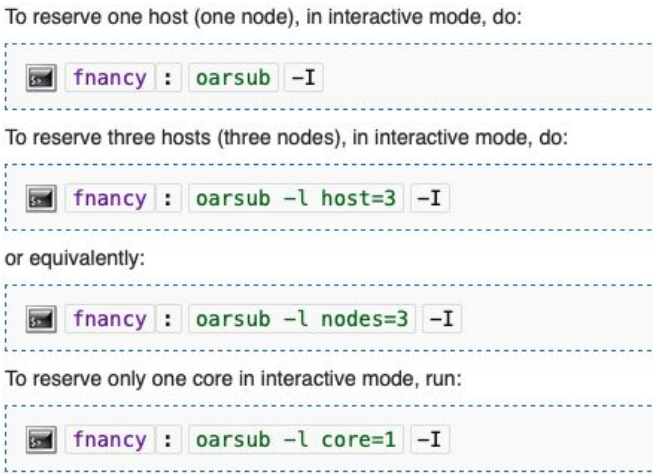

Gantt

Ganglia


Overview of Nancy @ 2020-10-18 23:50
Nancy Cluster Load last hour
Loads/Procs
7.0 k
6.0 k
5.0 k
4.0 k
3.0 k
2.0 k
1.0 k
0.0
23:00
23:20
23:40
1-min Now:323.6 Min:280.9 Avg:324.7 Max:371.
Nodes Now:182.0 Min:179.1 Avg:182.9 Max:186.
CPUs Now: 6.6k Min: 6.5k Avg: 6.6k Max: 6.
Procs Now:250.5 Min:196.1 Avg:315.5 Max:491.
Nancy Cluster Memory last hour
Bytes
25 T
20 T
15 T
10 T
5 T
0
23:00
23:20
23:40
Use Now: 797.4G Min: 750.1G Avg: 802.5G Max: 847.9G
Share Now: 0.0 Min: 0.0 Avg: 0.0 Max: 0.0
Cache Now: 390.5G Min: 382.0G Avg: 391.3G Max: 407.6G
Buffer Now: 10.1G Min: 9.8G Avg: 10.0G Max: 10.5G
Swap Now: 2.3G Min: 1.9G Avg: 2.0G Max: 2.3G
Total Now: 20.3T Min: 19.9T Avg: 20.2T Max: 20.7T
Nancy Cluster CPU last hour
Percent
100
80
60
40
20
0
23:00
23:20
23:40
User Now: 3.4% Min: 3.2% Avg: 3.6% Max: 4.4%
Nice Now: 0.0% Min: 0.0% Avg: 0.0% Max: 0.0%
System Now: 0.5% Min: 0.3% Avg: 0.6% Max: 1.4%
Wait Now: 0.0% Min: 0.0% Avg: 0.0% Max: 0.6%
Idle Now: 96.1% Min: 94.7% Avg: 95.8% Max: 96.4%
Nancy Cluster Network last hour
Bytes/sec
2.0 G
1.5 G
1.0 G
0.5 G
0.0
23:00
23:20
23:40
In Now:135.4M Min: 20.8M Avg:342.9M Max: 1.7G
Out Now:118.1M Min: 7.4M Avg:336.8M Max: 1.9G
RRDTOOL / TOBI OETIKER

Reserving resources with OAR

- OAR: resources and jobs management system (batch manager) in g5k
- Smallest unit of resource: core (cpu core)
  - E.g.: graffiti have 2 CPU with 8 cores/CPU, maximum reserved for 16 tasks
  - By default a OAR job reserves a **host** (=***nodes***, physical computer with all cpu/cores)
- Reservation syntaxe



Reserving resources with OAR: interactive mode

- Interactive mode
  - Use option `-I`
    - As soon as a resource is available, directly connected to that resource with an interactive shell. By default *walltime* = 1 hour
  - If you want to reserve GPU
    - `fnancy : oarsub -l gpu=1 -I -q production`
    - This means reserve 1 GPU with the associated cores in the queue production
    - Nodes with GPU are **exclusively** in the production queue in Nancy
  - Terminate reservation and return to frontend
    - exit or CTRL+d
  - Need more than 1 node or longer time (walltime):
    - `fnancy : oarsub -I -l nodes=2,walltime=0:30`

Reserving resources with OAR: passive mode

- Passive mode
  - By default, no need to add an option
  - Reservation in 2 steps
    - First reserve a node and ask it to sleep for a long time
    - Allocate a job_ID quickly
    - Then use this command to enter the host
  - Advantage: no worry about accidentally terminate your task (terminal closed or network disconnection)
  - More parameters:
    - -r: reserve a specific time in the future
-
- More options to reserve a resource check `oarsub --help`

```
fnancy : oarsub "sleep 10d"
```

```
fnancy : oarsub -C job_id
```

```
fnancy : oarsub -l nodes=3,walltime=3 -r '2020-12-23 16:30:00'
```

Job management

- View your list of jobs with `oarstat`
  - Option `-u` see only your jobs: `oarstat -u`
  - Option `-j job_id` see the state for this particular job
  - Status: W=waiting, L=launching, R=running, F=finish

```
Job id     Name           User           Submission Date     S Queue
---------- -------------- -------------- ------------------- - ----------
2758674                   cli            2020-10-18 19:17:56 R production
2759002                   cli            2020-10-19 09:38:00 R production
2759104                   cli            2020-10-19 10:28:41 R default
2759109                   cli            2020-10-19 10:30:43 R default
```

- Delete a job with `oardel`
  - `fnancy : oardel 12345`
- Passive mode jobs, **stdout** and **stderr** streams are created automatically
  - check out stream (or error stream) with `cat` at any time
  - `$ cat OAR.2758674.stdout`

Job management

- Specify the properties of host with option `-p`
  - exemples :
  - `fluxembourg : oarsub -p "cluster='granduc'" -l nodes=5,walltime=2 -I`
  - `flyon : oarsub -p "wattmeter='YES' and gpu_count > 0" -l nodes=2,walltime=2 -I`
  - oarsub also accepts SQL
- Extend the duration with `+time`:
  - `fnancy : oarwalltime 12345 +1:30`
  - Not whenever you want, check rules in Usage Policy

Some examples

- Ask for 1 core and launch a script called ‘my_script.py’
- Ask for 3 GPU in host ‘graffiti-4’ in site Nancy, queue production for 1 hour
- Ask for 20 cores in ‘grvingt’ in production queue and sleep 10 days
- Ask for 1 node in cluster ‘grvingt’ for 20 minutes, and launch script ‘run.sh’
- Check my reservations

- oarsub -ℓ core=1 "my_script.py --in $HOME/data/ --out $HOME/results/"
- oarsub -p "host in ('graffiti-4.nancy.grid5000.fr')" gpu=3,walltime=1 -q production
- oarsub -p "cluster='grvingt'" -ℓ core=20 "sleep 10d" -q production
- oarsub -p "cluster='grvingt'" -ℓ nodes=1,walltime=0:20 "bash run.sh" -q production
- oarstat -u

Customize software environment

Kadeploy

- `oarsub` gives access to resources configured in default environment
- Re-install the nodes with different software environment
  - Different Debian version, another Linux distribution, or even Windows
  - Can get root access to install the software stack
  - More detail ref this link
  - More about Kadeploy and `kadeploy3` commands, refer this link

Towards deep learning

Deep learning

- Creation of a virtual environment for python
- Installation of deep learning software
- Configuration of software (such as cudnn library, config file)
- Running DL software on Grid’5000
  - Reservation with oarsub
  - monitoring (log files, kill)
  - Use several GPU cards
- Tips and tricks, for detailed info follow this link

Deep learning - virtual env.

- Creation of a virtual environment for python
  - Go to Nancy g5k site
  - `inside : virtualenv /home/ login /venv`
  - Can precise interpreter with `-p` such as `--python=python3.7`
  - Activate virtual environment
  - `inside : source /home/ login /venv/bin/activate`
  - Otherwise, can do with anaconda

Deep learning - pytorch installation

- Pytorch
  - Reserve a cluster with GPU (graffiti, graphique, grimani, *etc.*)
  - In the host, install torch with pip or anaconda
  - Load module cuda and cudnn in current shell
    - `$ module av`
    - `$ module load cuda/11.0.1_gcc-8.3.0`
    - `$ module load cudnn/7.6.5.32-10.1-linux-x64_gcc-8.3.0`
  - Check if pytorch is correctly installed to work with GPU
    - `$ python3 -c "import torch; print(torch.cuda.is_available())"`

- Similar for Tensorflow

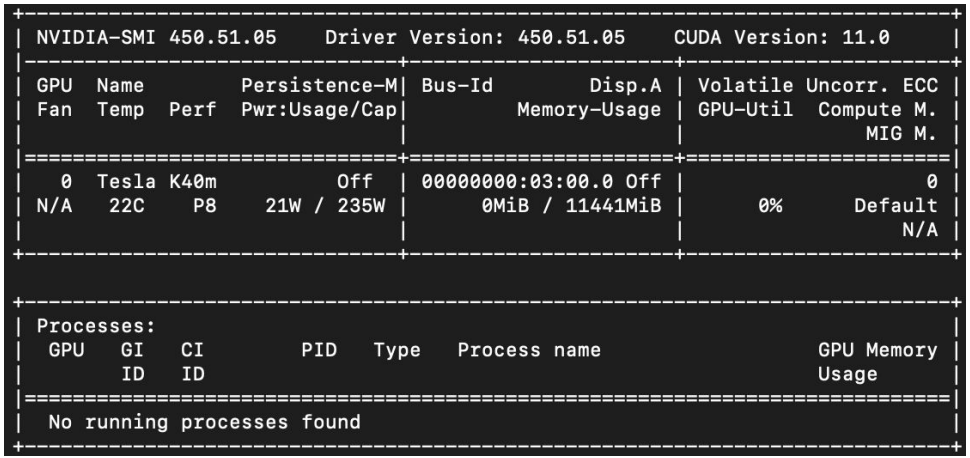
Deep learning - nancy site

- Available nodes
  - grimani: 6 nodes, each node has 2 Nvidia K40m GPU cards
  - graphique: 6 nodes, 2 x Nvidia Titian Black (graphique-1), 2 x Nvidia GTX 980 GPU (other nodes)
  - grele: 14 nodes, each node has 2 Nvidia Geforce 1080 Ti GPU cards
  - graffiti: 13 nodes, each node has 4 Nvidia Geforce RTX2080 GPU cards

- Each gpu cluster has 2 GPU cards
  - Script can use already the 2 cards
  - If want to use multiple GPU cards of one machine in parallel, ref this tuto

Deep learning - reservation

- Reserve one GPU
  - Interactive mode: `inside : oarsub -q production -l "nodes=1/gpu=1,walltime=0:20:00" -I`
  - Passive mode:
    - `inside : oarsub -q production -l "nodes=1/gpu=1,walltime=0:20:00" <path to a bash script>`
    - Move into the host: `site:~$ oarsub -C job_id`
  - Check GPU usage: `host:~$ nvidia-smi -l 2`



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.51.05    Driver Version: 450.51.05    CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K40m          Off  | 00000000:03:00.0 Off |                    0 |
| N/A   22C    P8    21W / 235W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Conclusion

Community

- Report the problems to the community
  - users@lists.grid5000.fr
- (if you want) join the technical committee
  - Subscribe to devel@lists.grid5000.fr
  - Discussions and bugs

Wrap up

We have seen

- Connecting to Grid’5000
- Infrastructure map, with some basic concepts
- Visualizing resources
- Transferring files
- Reserving resources with 2 modes
- Job management
- A deep learning framework

Wrap up

We have seen

- Connecting to Grid’5000
- Infrastructure map, with some basic concepts
- Visualizing resources
- Transferring files
- Reserving resources with 2 modes
- Job management
- A deep learning framework

We have used

- ssh
- site, cluster, node, core
- Gantt, Monika...
- scp, rsync
- oarsub
- oarstat, oardel, oarwalltime
- Pytorch installation

Wrap up

- Grid’5000 is a fantastic tool for your research
- Mastering it is challenging
- Be positive, find a problem, ask and share =)
- Questions?