Introduction : LGTC Shark cluster

Author : Michiel van Galen, Michel Villerius
Cluster admin : Michel Villerius
Updated : July, 13 2011
WiKi : http://shark.lumcnet.prod.intern/wiki

Configuration: CPUs Total:**240** , Nodes: **20**

Head node:

shark
- description : Rack Mount Chassis
- product : PowerEdge 2950
- vendor : Dell Inc.
- width : 64 bits
- CPU : 8 * Intel(R) Xeon(R) CPU L5430 @ 2.66GHz
- Memory : 16 GB
- Hard disk : Disk /dev/sda: 72.7 GB
  : Disk /dev/sdb: 1199.1 GB
- File-system : Hardware Raid 1 for sda
  Hardware Raid 0 for sdb

/ifs/exports/data	/data	nfs mount
/ifs/exports/home	/home	nfs mount
/ifs/exports/system	/share/isilon/system	nfs mount

Execution HOSTs:

3 * M600 Dell Blade server:
angelshark, blacktipshark, caribbeanshark
- description : Multi-system
- product : PowerEdge M600
- vendor : Dell Inc.
- version : PowerEdge M1000e
- width : 64 bits
- CPU : 8 * Intel(R) Xeon(R) CPU L5430 @ 2.66GHz
- Memory : 32 GB
- hard disk : 2 * 1TB 1 volume /dev/sdb
- File system : Hardware Raid 0 for sdb

/dev/sdb1	size=5.5G	/
/dev/sdb4	size=1.8T	/tmp
/dev/sdb3	size=942M	/var

7 * M610 Dell Blade server:
dogfishshark, greatwhiteshark, hammerheadshark, lemonshark, megamouthshark, tigershark, whaleshark
- description : Multi-system
- product : PowerEdge M610
- vendor : Dell Inc.
- version : PowerEdge M1000e
- width : 64 bits
- CPU : 12 * Intel(R) Xeon(R) CPU X5670 @ 2.93GHz
- Memory : 96 GB
- hard disk : 2 * 1TB /dev/sda /dev/sdb
- file system : Software raid0 only for /tmp

/dev/sda1	size=5.1G	/
/dev/sda3	size=888M	/var
/dev/md0	size=1.8T	/tmp

7 * M610 Dell Blade server:
epauletteshark, frilledshark, kitefinshark, nightshark, pygmeshark, threshershark, zebrashark

description	: Multi-system
product	: PowerEdge M610
vendor	: Dell Inc.
version	: PowerEdge M1000e
width	: 64 bits
CPU	: 12 * Intel(R) Xeon(R) CPU X5670 @ 2.93GHz
Memory	: 64 GB
hard disk	: 2 * 600GB /dev/sda /dev/sdb
file system	: Hardware raid0

/dev/sda1	size=5.1G	/
/dev/sda5	size=844M	/var
/dev/sda6	size=1.1T	/tmp

1 * M610 Dell Blade server:
makoshark

description	: Multi-system
product	: PowerEdge M610
vendor	: Dell Inc.
version	: PowerEdge M1000e
width	: 64 bits
CPU	: 12 * Intel(R) Xeon(R) CPU X5670 @ 2.93GHz
Memory	: 128 GB
hard disk	: 2 * 600GB /dev/sda /dev/sdb
file system	: Hardware raid0

/dev/sda1	size=5.1G	/
/dev/sda5	size=844M	/var
/dev/sda6	size=1.1T	/tmp

1 * M910 Dell Blade server:
baskingshark

description	: Multi-system
product	: PowerEdge M610
vendor	: Dell Inc.
version	: PowerEdge M1000e
width	: 64 bits
CPU	: 24 * Intel(R) Xeon(R) CPU E7540 @ 2.00GHz
Memory	: 256 GB
hard disk	: 2 * 600GB /dev/sda /dev/sdb
file system	: Hardware raid0

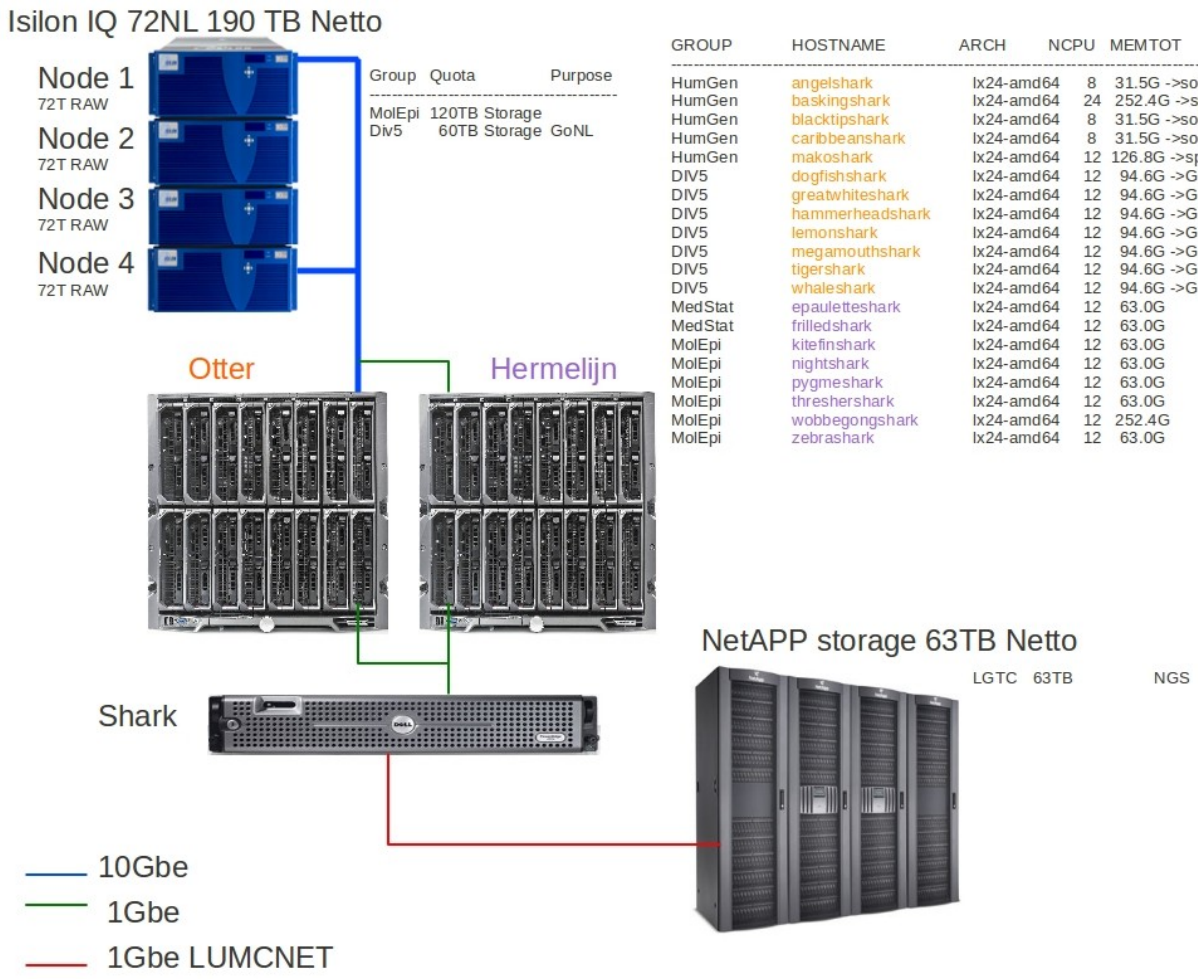
/dev/sda1	size=5.1G	/
/dev/sda5	size=844M	/var
/dev/sda6	size=1.1T	/tmp

1 * M910 Dell Blade server:
wobbegongshark

description	: Multi-system
product	: PowerEdge M610
vendor	: Dell Inc.
version	: PowerEdge M1000e
width	: 64 bits
CPU	: 12 * Intel(R) Xeon(R) CPU X7542 @ 2.67GHz
Memory	: 256 GB
hard disk	: 2 * 600GB /dev/sda /dev/sdb
file system	: Hardware raid0

/dev/sda1	size=5.1G	/
/dev/sda5	size=844M	/var
/dev/sda6	size=1.1T	/tmp

Graphical representation of the SHARK cluster



GROUP	HOSTNAME	ARCH	NCPU	MEMTOT
HumGen	angelshark	lx24-amd64	8	31.5G ->solexa/helicos
HumGen	baskingshark	lx24-amd64	24	252.4G ->special Hiseq
HumGen	blacktipshark	lx24-amd64	8	31.5G ->solexa/helicos
HumGen	caribbeanshark	lx24-amd64	8	31.5G ->solexa/helicos
HumGen	makoshark	lx24-amd64	12	126.8G ->special R/matlab
DIV5	dogfishshark	lx24-amd64	12	94.6G ->GoNL
DIV5	greatwhiteshark	lx24-amd64	12	94.6G ->GoNL
DIV5	hammerheadshark	lx24-amd64	12	94.6G ->GoNL
DIV5	lemonshark	lx24-amd64	12	94.6G ->GoNLNGS
DIV5	megamouthshark	lx24-amd64	12	94.6G ->GoNL/NGS
DIV5	tigershark	lx24-amd64	12	94.6G ->GoNL/NGS
DIV5	whaleshark	lx24-amd64	12	94.6G ->GoNL/NGS
MedStat	epauletteshark	lx24-amd64	12	63.0G
MedStat	frilledshark	lx24-amd64	12	63.0G
MolEpi	kitefinshark	lx24-amd64	12	63.0G
MolEpi	nightshark	lx24-amd64	12	63.0G
MolEpi	pygmeshark	lx24-amd64	12	63.0G
MolEpi	threshershark	lx24-amd64	12	63.0G
MolEpi	wobbegongshark	lx24-amd64	12	252.4G
MolEpi	zebrashark	lx24-amd64	12	63.0G

Data Protection Isilon Storage

Isilon has an OneFS file system that includes a FlexProtect technology. This technology stores protection information for each file independently and distribute this protection information over the complete file system. Here at the LUMC we have set the protection level to 1:2, 1 node and 2 disks can fail without data loss.

This document will very briefly introduce you to the scheduling system that we use on the shark computer farm. It also assumes you know your way around in the terminal and that you will use the farm in a responsible way and respect the available resources.

Guidelines

- Do not permanently store data on the Shark cluster, we take no responsibility for loss of your data.
- Do not use the /tmp directory for storage, this directory is used for SUN GRID ENGINE and Helicos analysis.
- Do not use your home folder for large files. Create a directory in your own network share ”groupdir” and work from there.
  - **Network shares :**
  - /data2 ,917G. Only For Shark Admin members.
  - /Solexa-storage/Solexa-storage_01 ,11T. Only For LGTC Solexa members
  - /Solexa-storage/Solexa-storage_02 ,11T. Only For LGTC Solexa members
  - /Solexa-storage/Solexa-storage_03 ,11T. Only For LGTC Solexa members
  - /Solexa-storage/Solexa-storage_04 ,10T. Only For LGTC Solexa members
  - /Solexa-storage/Helicos-storage ,10T. Only For LGTC Helicos members
  - /DATA-temp ,10T. Only for LGTC members.
  - /data/MolEpi ,120T. Only For MolEpi members
  - /data/DIV5, 60T. Bought for the GoNL project, but groups who do not have storage for calculations on the Shark cluster can use their group folder for calculations but DO NOT store data here.
- Do not produce a heavy work load on the head node.
- Do not run tests on the SHARK cluster. This is a production cluster. Run test on your own workstation.
- Do not execute programs on the head node.
- Do not change your public and private key on this cluster located at ~/.ssh
- Do not share your login credentials with others.
- Always qsub your jobs.
- Software will only be updated once every 6 months, Keep in mind that this is a production server with lots of users and that we cannot update the software for every single user as he or she wishes.
- New software can be requested, but this software will be evaluated if this software can and may run on this SHARK cluster and therefore will take at least 4 weeks for a decision.
- Jobs with a know run time longer than 24hr, should not fill up the complete cluster, run those jobs with respect for other users.
- Jobs that take up all resources (memory, disk space, temp disk space etc.) Can and will be deleted by the admin without notice.
- When you encounter a problem please use our Trac bugtracking system for issues and/or feedback.
- These rules and guidelines can be changed anytime by the cluster admin.

Violation of these guidelines or other irresponsible use will result in your account being suspended!

Remember that a cluster runs many different processes submitted by many different users. If you are not sure what you are doing, ask your administrator before submitting very intensive jobs! For more examples and useful commands you can check '/data/Scripts/SGE-howto/' on the shark.

Mount points and directories:

NFS shared directories available from head node and execution nodes

Group	Disk Space	Storage
MolEpi	120TB	Isilon
Div5	60TB	Isilon
LGTC	63TB	NetApp

Mount Point	**Storage**	**Size**	**Usage**
/home	Isilon	1TB	DO NOT store large DATA here, only your scripts!
/usr/local	Isilon	1TB	where all the programs are located that needs to be executed, if you miss something ask the cluster admin, only he can add new programs or versions there.
/data	Isilon	178TB	Divided in a Directory MolEpi and Div5, only MolEpi can store their date (120TB) in /data/MolEpi, /data/DIV5/GoNL (60TB) is for the GoNL project. Space what is not needed for storage of the GoNL project can be used for calculating on the Shark cluster by others that do not have a storage. /data/div5/... is for direct calculating on the Shark cluster, data may be stored there for a **short period** of time (max. the time it takes to run your jobs that needs this data).
/data/MolEpi	Isilon	120TB	Storage space only for the MolEpi group
/data/DIV5/GoNL	Isilon	60TB	Storage space only for the GoNL project
/data/DIV5/...	Isilon	none	Space that is not in use by the GoNL project can be used to calculate on the Shark cluster, DO NOT abuse this space, use data only for calculating purpose and remove the data and results afterwards. Move results to your own dedicated storage, data here can and will be removed by the admin if storage is needed for the GoNL project, do not leave the data there for more time then your job needs to run.
/DATA-temp	NetApp	10TB	Directory for temporarily storage of data that needs to be analyzed, only for LGTC members.
/Solexa_storage/..	NetApp	53TB	Storage mount point for the LGTC Next Generation Sequencing, can only be accessed by the LGTC group members

The Shark cluster (on it's own) is NOT for DATA STORAGE! Put your data that needs to be analyzed in the appropriate place (like /tmp on the execution nodes, do not use /tmp of the head node). Results need to be transferred to the appropiate storage (/data/... or /Solexa-storage/.. for some groups, and others need to use their own storage) and all the initial data needs to be removed. If one of the Hard disks breaks and there is data loss we are not responsible. There are no backups for the local disk of the Shark cluster. Also disks can be purged of data anytime without any announcement.

Queues

A queue is needed when submitting jobs, if no queue is given the SGE submits your job to your default queue. There are several queues available. Only the administrator can create others if needed.

```
vill@shark:~$ qstat -g c
DIV5ngs.q
LGTC.q
LGTC_HiSeq.q
all.q
helicos.q
velvet.q
```

Batch Queuing Commands

The following commands give an overview of the most important SGE commands please read the sge_intro manual with the command : `man sge_intro`

qsub	Submit a job to the queue
qdel	Cancel a queued or running job
qhold	Place a queued job on hold
qstat	Check the status of queued and running jobs
qstat -g c	List all valid queue names
qstat -u "*"	List all jobs for all users
qalter	Change parameters for job (give job id) waiting in queue

Brief examples:

You can login to the Shark cluster head node with your user name and password provided to you by the system administrator. From there you can submit your jobs (qsub) or log in to an individual node (qlogin). Both methods will be managed by the scheduling system which will distribute the available resources.

To submit jobs, we use the qsub command. The qsub command requires a file(script) which describes what needs to be run in what way. We also need a script that we want to submit.

The example script that we want to execute, saved in this example as my_first_job.sh:
You can also use svn to get the example scipts.

```
cd ~
svn checkout https://www.mutalyzer.nl/svn/shark/
```

Once your svn repository is created you only have to update your svn repository.
while in your svn directory:

```
svn update
```

The scrips are located in the directory ~/shark/http://www.google.nl/trunk/

```
#!/bin/bash
echo 'Starting job...'
sleep 10
echo '10 seconds, end of script.'
```

qsub example file, saved in this example as run_my_first_job.sh:

```
#!/bin/bash
#$ -S /bin/bash
#$ -q all.q
#$ -N my_first_job
#$ -cwd
#$ -j Y
#$ -V
#$ -m be
#$ -M email@address.lumc

echo Start time : `date`
/home/user/my_first_job.sh
echo End time : `date`
```

Every line starting with "#$" is a parameter to the SGE.
The options explained:
-S = Used to define the shell
-q = The 'sub cluster' that your job will go to (use all.q unless the admin tells you otherwise)
-N = Your job name (can not start with a number)
-cwd = Used to let the output be put in the Dir you submitted the job from
-j Y = The standard error of the batch job and The standard output of the batch job are joined together
-V = Specify that all of the environment variables of the process are exported to the context of the batch job.
-m be = mail when job starts (b = begin) and when the jobs ends (e = ends) can also be: -m e
-M = the email address where the info is send to
With a script to run and a script to submit we can submit our job as:

```
qsub ./run_my_first_job.sh
```

```
Your job 1517 ("my_first_job") has been submitted
```

Your job will get a number which you need to track the progress, errors, or for canceling. Once submitted you can check the status of your job with the qstat command:

```
qstat
```

```
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
517     0.00000 my_first_j chiel        qw    06/09/2010 13:24:15                                    1
```

Below 'state' you can read 'qw' which means queue waiting. Once the head node finds available resources this will change to 'r' which means running.

Some Sun Grid Engine SGE state letter symbol codes:
qw pending
hqw pending hold
r running
t transferring
s job suspended
Eqw Error
d deletion

With qhost you can check the memory usage on the different nodes:

```
qhost
```

```
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
angelshark              lx24-amd64      8  2.11   31.1G    1.4G    7.5G   30.5M
blacktipshark           lx24-amd64      8  0.00   31.1G  121.4M    7.5G   44.3M
caribbeanshark          lx24-amd64      8  0.01   31.1G  137.5M    7.5G   42.5M
```

To delete your job you can use qdel.

```
qdel 1517
```

```
chiel has deleted job 1518
```

If your job creates files they will be put in the working directory unless told otherwise. In this example the script only prints something to screen. If you don't catch this output (by adding a >output.txt after the invocation of your script) this has to go somewhere.

For each job an error and output file will be generated if the option #$ -j Y is not given. You can find those files in

the directory you ran your scripts from (if “#$ -cwd” was included in the submission script). Typically the filename will include your job name and number. The contents of theses files are the standard output and error of your submitted script and everything that the submission script may have printed to the screen. In this case the files are called as follow:

```
my_first_job.o1517
my_first_job.e1517
```

Directly logging in to a node is possible with the command qlogin.

```
qlogin -q all.q

Your job 4952 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 4952 has been successfully scheduled.
Establishing built in session to host blacktipshark.cluster.loc ...
chiel@blacktipshark:~$
```

This will open a connection to a node reserving it until you exit. You can directly run jobs in the console. The complete node will scheduled for your session. To avoid the cluster being overloaded with idle qlogin sessions your session will automatically logout after being idle for 20 minutes.

Directly logging in to a specific node is possible with the command:

```
qlogin -q all.q@tigershark

Your job 20192 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 20192 has been successfully scheduled.
Establishing built in session to host tigershark.cluster.loc ...
vill@tigershark:~$
```

What if your job does not run ?

If your job shows "Eqw" or "qw" state when you run qstat. Check your job with:

qstat -j <job number>

Did your job ran but something else isn't working, get more info with:

qacct -j <job number> (check lines with "failed" and/or "exit_status")

If you have an "access denied" message somewhere in your job explanation, you probably have a permission problem. Your user account does not have the privileges to read from/write to where you told it (this happens with the -e and -o options to qsub often). Check if you can write to the specified directories.

To avoid permissions problems, cd into the directory on the NFS where you want your job to run, and submit from there using qsub -cwd to make sure it runs in that same directory on all the nodes.

Not a permissions problem? Maybe the nodes or the queues are unreachable. Check this with:

qstat -f or qstat -F