Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO TUTELADO AISI
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Despregue dun cluster virtual para HPC con Qlustar

Estudante: Mauro A. de los Santos Nodar
Profesor: Xoán Carlos Pardo Martínez

A Coruña, maio de 2020.

Índice Xeral

Índice de Figuras

Capítulo 1

Primeiros pasos

1.1 Instalación de Qlustar Operating System

O primeiro paso de todos para despregar o noso *cluster* virtual para *HPC* será ter listo o sistema operativo ***Qlustar***. Para isto, iremos na súa documentación ao apartado de instalación [1] e procederemos a seguir os pasos detallados. A continuación, expóñense de forma resumida os máis importantes:

- Primeiro de todo descargaremos a imaxe máis actual do SO en [2]
- Mentres se descarga configuraremos unha máquina virtual onde teremos o noso *Head Node*. Mencionar que no noso caso a plataforma utilizada para crear esta máquina será un **Virtualbox v6.0.16**, a cal terá as seguintes características:
  - 1 disco virtual dinámico de 160 GB
  - 5 GB de RAM (2 GB por ter o *Head* cun *FE* como veremos máis adiante, e 1.5 máis por cada nó de cómputo)
  - Soporte para virtualización, elixindo a opción de *KVM* en paravirtualización, xa que é a única infraestrutura probada oficialmente por *Qlustar*
  - Dous adaptadores de rede, un deles con conectividade a Internet en modo *bridge*, e outro en modo *internal*

Unha vez feito isto, teremos unha máquina virtual cos requisitos necesarios para *Qlustar* e unha imaxe do mesmo, polo que solo quedará comezar co proceso de instalación do SO seguindo as instruccións dispoñibles na documentación. A continuación, coma no caso anterior, coméntanse as máis relevantes:

- Tras insertar a imaxe como disco óptico e establecela como primeira opción de arranque, seleccionaremos a primeira opción: *Qlustar Installation*, onde comezaremos ca instalación. O primeiro serán axustes básicos triviais como a zona horaria, o teclado, etcétera.

- En canto á configuración de disco, poderemos elixir en que disco instalar *Qlustar*, no noso caso, no único de 160 GB dispoñible, aínda que se houbera varios, *Qlustar* darianos a opción de configurar diferentes *RAIDs*. Tamén elixiremos onde instalar o */home filesystem*, no mesmo volume ou de forma externa e acceder por *NFS*, por exemplo. Imos ca primeira opción para acotar a extensión e complexidade do traballo.

- Axustes de rede, paso fundamental, tanto para xestionar a *IP* do *cluster*, como para as *IP's* internas aproximadas que serán necesarias, introducindo o número de nós, no noso caso 2. Tamén poderemos cambiar o *hostname* e configurar un *mail*, importante para as mensaxes de monitorización. En xeral, como ben se di na documentación, **se non hai razóns concretas para cambiar a configuración por defecto, recoméndase non facelo**, así que nós imos deixalo todo por defecto. O importante deste paso recae en **diferenciar as dúas interfaces de rede**, unha para a conexión a Internet e outra para a rede interna do *cluster*, por eso a importancia de ter dúas interfaces de rede activas na máquina.

- O segundo paso fundamental deste punto é a configuración necesaria para separar as actividades de usuario das actividades do sistema, para así mellorar o rendemento. Primeiro de todo, teremos que elixir se o noso nó *Front End* queremos que sexa unha máquina virtual correndo no noso *Head* (opción **FE VM**) ou unha máquina física externa. Aínda que se recomende esta última, por motivos de rendemento e modularidade, nós imos elexir a opción de que a nosa **Front End sexa unha VM no noso Head**.

- Por último temos as *Edge Platorms e as Package Bundle Selection* para elixir un *cluster* baseado en **Ubuntu/Bionic** no noso caso, e en canto aos paquetes, como vimos na teoría da materia, haberá que instalar os paquetes xustos e necesarios polo que instalaremos só **Slurm**, o cal usaremos ao longo deste traballo para a *HPC*. Estes paquetes aquí elexidos instalaránse só no *Head node*.

Chegados até aquí, temos a primeira parte da instalación e creación do *Head Node* feita, agora quedará reiniciar e acometer a última.

1.2 Finalización da instalación e creación do Head-Node

A primeira vez que *booteemos* o noso *Head* teremos que facer a configuración inicial para poñelo en marcha. Isto, faráse mediante a execución do seguinte comando, despois de loguearse como *root* ca *password* que indicamos na instalación do paso anterior:

```
1 #/usr/sbin/qlustar-initial-config
```

Neste segundo proceso de configuración, as principais accións a facer son:

- Introducir o nome do noso *cluster* virtual a crear.
- Configurar a *NIS database*, onde deixaremos como dixemos antes, as opcións por defecto, e establecer os contrasinais de *Nagios*, dos accesos *SSH*, de *Qluman*, de *Slurm* e de *MariaDb*.
- Crear unha conta de proba que serverá para explorar o *cluster* virtual que eliximos crear durante a instalación.

Con isto, finalizariamos esta segunda etapa de instalación, polo que só quedaría facer un *reboot* final onde despois del, xa teriamos correndo o noso cluster cun ***Head-Node*** e co seu ***Front End (FE)*** (correndo como unha *VM* no propio *Head*). Teriamos tamén preconfigurados un número de *nós Demo VM* e estariamos listos para desplegalo e facer as primeiras probas.

Capítulo 2

Posta en marcha do Head-Node

2.1 Primeiros pasos no Head-Node

2.1.1 Arrancar o noso Cluster Virtual

O primeiro que faremos será arrancar o noso **Cluster-Demo virtual** creado no capítulo anterior, para facer un *testeo* do mesmo, co comando:

```
#demo-system-start
```

Se queremos cambiar a configuración ca que este arranca, só teremos que editar o ficheiro **/etc/qlustar/vm-configs/demo-system.conf**, onde están as configuracións por defecto dos *Cluster Demo*. Tamén poderemos accceder ás *shells* das diferentes partes do *Cluster* co comando:

```
#console-[fe/demo]-vms
```

Por último, para *checkear* o correcto estado dos nós bastaría con facer un:

```
#dsh -a uptime
```

2.1.2 Primeiros pasos con Qluman

Unha vez co *Cluster* operativo, teremos que familiarizarnos co xestor do mesmo, ***Qluman***. Polo que imos inicialo seguindo os seguintes pasos:

- Ao usar un sistema de chaves pública/privada, teremos que xeralas e intercambialas para telo operativo, aínda que isto en pasos posteriores se poderá facer de forma gráfica, para o primeiro *login* necesitamos facelo por consola. Primeiro de todo executaremos:

```
    #qluman-cli --gencert -u admin -o token

```

- Unha vez temos o *token*, instalaremos a ***GUI de Qluman*** con

```
#apt install qluman-qt
```

  O problema ven en onde instalala, xa que é pesada e consume recursos. A documentación recoméndanos instalala noutra máquina, por exemplo no noso *PC*, ao que él chama *User Workstation*. Aínda que tamén poderiamos instalala no *Head Node* e lanzala por *X11* de forma externa. Por motivos de rapidez e de extensión do traballo imos facelo desta última maneira. Polo que despois de instalalo no *Head* co comando anterior, para acceder, dende outra estación, por exemplo o ordenador persoal, estando ambos na mesma *LAN*, executamos:

```
#ssh -X root@servername qluman-qt
```

  Con isto teremos o primeiro paso con *Qluman* feito, polo que en capítulos posteriores intentaremos explotar ao máximo o seu potencial, pero de momento, simplemente comprobamos que nos podemos *loguear* e acceder á pantalla de inicio de *Qluman*.

- Para acabar esta fase de primeiros pasos no *Head*, crearemos un usuario de proba, de forma sinxela co script *adduser*, o cal usaremos posteriormente para a execución das probas *HPC* no noso *cluster*:

```
#adduser.sh -u test -n "Test User"
```

2.2 Probando o noso cluster

Xa temos o noso ***HPC Cluster*** con **2 nós-demo** virtuais creados, un **Head node** e un **Front Ent node** correndo como VM no *Head*, polo que estamos listos para probar o seu funcionamento. Isto farémolo de dúas maneiras:

2.2.1 Compilando un programa MPI

Primeiro comprobaremos que a infraestrutura *OpenMPI* está ben instalada e operativa executando dous sinxelos programas en *C* e *Fortran*:

```
#mpicc.openmpi-gcc -o hello-world-c hello-world.c
#mpif90.openmpi-gcc -o hello-world-f hello-world.f90
```

Despois disto, se todo foi ben, teremos dous executables que serán cos que *testearemos* a infraestrutura usando *Slurm* da seguinte forma:

```
#salloc -N 2 --ntasks-per-node=2 --mem=20 -p demo srun
hello-world-c
```

```
#sbatch -N 2 --ntasks-per-node=2 --mem=20 -p demo
hello-world-f90-slurm.sh
```

Tras executar estes comandos e ver a saída pertinente, poderemos observar se o *Cluster* se comportou de forma exitosa, usando as tarefas, procesos e nós *printeados*.

2.2.2 Probando o Linpack Benchmark

Probemos agora con algo que non sexan *Hello Worlds*, co ***Linpack Benchmark***. Este *benchmark*, como vimos en teoría, é un dos máis coñecidos no mundo do *HPC* polo que é moi interesante e útil probalo neste traballo, a parte de que nos servirá para poñer a proba o noso *cluster*.

Qlustar xa ven cunha versión precompilada deste e ca configuración necesaria para facilitar a súa proba. Executaremos, no noso caso que temos 2 nós, o seguinte comando:

```
#linp-2-demo-nodes
```

Podemos comprobar con **Squeue** que o traballo comezou, e facer un *SSH* a calquera dos nós para ver que temos os procesos adecuados correndo, facendo un *top* por exemplo.

Podemos ver as saídas deste *benchmark* en **$HOME/bench/hpl/run/job-<jobid>-*/openblas/job-<jobid>-*-<run>.out** onde o *jobid* será o devolto polo *Squeue* anteriormente dito e o *run* será un enteiro.

Da forma que ven preconfigurado este *benchmark* estaráse executando nun bucle infinito polo que unha vez comprobado como ata agora o seu correcto funcionamento, poderemos detelo con **Scancel *jobid***.

Todos os comandos de *Slurm* usados nestas probas, serán explicados e tratados en profundidade en capítulos posteriores da memoria.

2.3 Recapitulamos

Unha vez aquí, temos un ***Cluster HPC*** totalmente configurado e probado con diferentes *testbenchs*, con 2 nós *demo* virtuais de cómputo e un nó *Head* co seu *Front* virtualizado. Temos diferente ***software*** instalado para diferentes obxectivos e a posibilidade de moldear esta infraestrutura ao noso gusto. Tamén fixemos as nosas primeiras probas con **Slurm**, executando os diversos comandos expostos anteriormente, e tamén temos a **Qluman** listo para ser usado. Polo que agora, a primeira tarefa a acometer despois disto, será crear nós que non sexan tipo *demo*.

Capítulo 3

Cluster con nós *reais* usando Qluman

3.1 Creando os novos nós

Para este apartado usaremos **Qluman**, o cal foi instalado e posto en marcha no apartado anterior. Mediante este xestor gráfico, as tarefas faránse moito máis intuitivas e doadas, e poderemos chegar ao obxectivo deste apartado dunha forma rápida e sinxela. Imos ver entón, como engadir agora, a un *HPC Cluster*, compoñentes **non** *demo ou virtuais*. Comezamos:

- Para poder comezar a usar *Qluman* de forma axeitada, o primeiro que faremos será importar o *token* que xeramos en apartados anteriores. Unha vez feito isto, teremos xa acceso ao noso *Head Node* e ao *cluster* en xeral, onde poderemos ver o estado de todas as compoñentes ao completo, así como configurar novos elementos, cambiar axustes, etcétera. Pero primeiro, imos co obxectivo deste apartado, os **nós NON Demo**.

- Unha vez conectados ao *cluster*, imos á pestana de *Manage Hosts*, *Add Hosts*. Aquí poderemos engadir ao noso *cluster* xa existente os nós que queiramos e do tipo que queiramos. Para probar os que nos quedaron no tinteiro, imos crear varios ***Standard Nodes***, nós estándar ou *reais* de cómputo. Para iso, o *hostname* e a *IP* interna no *cluster* será automaticamente asignada por *Qlustar*, nós só nos teremos que encargar da **MAC**, que poderemos facelo de dúas maneiras:

  - Especificándoa manualmente
  - Importando un ficheiro de texto cunha definiicón de diversas *MACs*, automatizando así o proceso, podendo dicir incluso en que parte do *cluster* ubicar estes nós.

  Unha vez que engadimos os necesarios, *clickamos* en aceptar e xa teriamos os nosos nós estándar listos para manexar dende *Qluman* coma o resto. Como ben se di na configu-

ración, tan só con estes pasos e especificando a plantilla de *Qlustar* a usar para o nó (*FE VM, Standard, Head, Demo VM*) teriamos todo listo para ter o noso novo *host* operativo.

3.2 Configurando os nós

Con todo isto, poderemos configurar ao noso antollo estes *hosts* creados. Isto faráse, coma todo este capítulo, dende *Qluman*, e estará dividido en diferentes apartados. Aínda que a documentación é moi extensa e precisa cos diferentes tipos de configuración e parámetros a tocar, imos resumir ou destacar os fundamentais que son:

- As ***Global Templates***, configuracións aplicábeis a todo o *cluster* en conxunto.
- As **Host Templates**, que son configuracións asignadas individualmente a cada *host* ou a grupos de *hosts* do mesmo tipo.
- A **configuración específica** de cada *host* individual, onde poderemos cambiar as características concretas de cada un (como pode ser a cantidade de *RAM*, disco, etcétera)

Este proceso de configuración pode ser moi longo ou moi sinxelo dependendo de onde queiramos chegar, pero non deixa de ser un proceso totalmente sistemático, basado na *GUI de Qluman* e onde simplemente se van indicando os parámetros a cambiar de cada *cluster*, *host* ou grupo deles.

3.3 Explotando Qluman

Polo que unha vez cos nós estándar incorporados ao noso *cluster* e configurados de forma axeitada, para acabar este capítulo, destacaremos a cantidade de opcións e posibilidades de configuración que nos ofrece *Qluman* e intentaremos expoñer algunhas das que poden chegar a ser máis útiles ou frecuentes:

- **Execución de comandos**, tanto en nós individuais coma en grupos deles. Dunha forma moi sinxela, poderemos executar os comandos que queiramos dende esta interfaz gráfica de forma simultánea nos nós que prefiramos, o que sen dúbida, será unhas das funcionalidades de *Qluman* máis habituais. Bastará con seleccionar os nós sobre os que traballar, *Execute* e elexir ou escribir o comando. Poderemos ver a saída en tempo real, a saída final e o estado da execución gracias á potente interfaz gráfica desta tarefa.
- **Xestión de usuarios e roles**. Dende *Qluman* poderemos crear, eliminar e editar á medida das necesidades do noso *cluster*, todo tipo de accións relacionadas con permisos de maneira moi sinxela na pestaña *Manage Cluster*, *Manage User/Roles*.

- **Log Viewer**. *Qluman* dános unha gran infraestrutura para a análise dos *logs*, podendo acceder a eles na pestana *Messages* a través do botón da pantalla inicial abaixo á dereita, para despois aplicar multitude de filtros, en base á prioridade, orixe, categoría, *strings* concretos, etcétera.

- E **moito máis**. Realmente dende *Qluman* podemos facer practicamente a totalidade da xestión do *Cluster* e dos seus nós. Un bo seguimento en tempo real do estado de toda a infraestrutura e unha boa auditoría a través de *logs*. Podemos ampliar á nosa medida o *cluster* e incluso crear outros novos. Unha vez chegamos a este punto, queda na man do lector explorar *Qluman* e sacarlle o maior partido posible.

Polo tanto, temos xa o noso *cluster* con nós virtuais e reais, *testeado*, operativo e configurado gracias á axuda de *Qluman*, co cal nos familiarizamos e do que vimos gran parte das súas capacidades. Polo que agora, queda afondar no mundo do *HPC* e sacarlle partido á infraestrutura despregada, e isto, farémolo co uso de **Slurm**, como veremos no seguinte capítulo.

Capítulo 4

Slurm para afondar no cluster HPC

Slurm dividiráse nunha serie de comandos que explotarán o *cluster* que teñamos tras realizar os pasos anteriores aplicando *HPC*. Polo que imos ver os máis importantes e analizar as súas capacidades e opcións posíbeis:

4.1 Sbatch

Funciona, coma todos os comandos *Slurm*, en base a *flags*, onde podemos destacar:

- **-N**: Número de nós para o traballo ou *job*
- **-n**: Número de tarefas a ser executadas
- **–ntasks-per-node**: Número de procesos a correr en cada nó
- **-c**: Número de *CPUs* requeridas para cada tarefa
- **–memory**: Memoria requerida para o *job* a executar, se este excede o valor, termínase a execución
- Para xestionar a asignacion de *GPU* temos o *flag* **–gres=gpu:<gpu_name>:<number of gpus>**. Aínda que isto será moi específico do *cluster* en cuestión, dependendo das particións *GPU* que teñamos, etcétera
- **–time**: Para asignar *walltimes* ás tarefas e que se pasado este tempo non finalizaron, sexan terminadas
- **–mail-type e –mail**: Para indicar cando notificar a un usuario (cando comece o *job*, cando se chegue a un tempo determinado ou cando se acabe) e a qué *email* notificar.

4.2 Srun e salloc

Aínda que tarefas como compilar ou editar ficheiros adoitan estar limitadas aos nós *Front*, cando van ser acometidas en longas sesións, poden consumir recursos de nós cómputo con estes comandos. Adoitan tomar os mesmos *flags* que *sbatch*, e un dos seus usos podería ser, en primeiro lugar, establecer os recursos necesarios con *salloc* e despois na *shell* devolta polo mesmo, executar un *srun* para acometer as tarefas.

4.3 Sinfo, Squeue e Sacct

Os seguintes comandos poderiamos clasificalos nun apartado de **monitorización e *accounting*** dos nosos jobs:

- **Sinfo** mostrarános os *walltimes* e propiedades máis importantes dos nosos *clusters*
- **Squeue** darános tanto os *walltimes* coma os tempos de execución de todas as tarefas que executemos con *sbatch*
- **Sacct** mostrarános todos os *jobs* executándose e o seu estado. Permite tamén filtrar por este último e por tempo de execución, así como facilita obter diversas métricas dos *jobs* coma enerxía, uso de memoria, *CPU*, etcétera.

4.4 Scancel

Elimina traballos comezados con *sbatch*. Ten multitude de opcións e filtros para, por exemplo, eliminar traballos por estado, usuario, nós, identificador, etcétera.

4.5 Scontrol

Serve para modificar *en quente* diferentes parámetros asignados a *jobs* que están correndo para, entre outras cousas, modificar o seu *walltime* ou as súas dependencias.

4.6 Arrays e Chains Jobs

Para agrupar *jobs* e que sexan tratados coma unha unidade haberá que usar a opción –**array** xunto co número de índices a ter.

Por outro lado, para crear dependencias entre *jobs* e xerar as coñecidas *Chain Jobs* teremos que usar o flag –**dependency**, onde indicaremos o *job* que necesita ser finalizado para que comece o traballo seguinte.

Capítulo 5

Probando os comandos Slurm

Unha vez vistas as principais partes e funcionalidades dos diversos comandos de *Slurm*, imos probar algún deles e mostrar exemplos da súa execución e saída:

Sbatch

Todos os *Sbatch* daránnos unha saída similar, indicando que o traballo foi *presentado* ao *cluster* e *printeando* seu *jobid* correspondente. Vemos, en primeiro lugar, uns exemplos desta típica saída:

```
 $ sbatch --job-name=test --begin=2020-01-01T08:00:00 ./naga.1.sh
Submitted batch job 1405163
```

Figura 5.1: Sbatch

```
[zhuz@cdl5:~/ksl/training/byKSL/20150520/vasp/01]$sbatch z_jobs_shaheenII
Submitted batch job 9668
```

Figura 5.2: Sbatch (2)

Nestas imaxes, vemos tamén como a execución consiste nun só comando cas *flags* que queiramos indicadas, como explicamos no apartado anterior, pero tamén é interesante outra forma de usar **Sbatch**, como podería ser, en primeiro lugar escribir un *script* indicando os *flags* necesarios, da seguinte maneira:

```
[zhuz@cdl2:/scratch/tmp/zzy]$cat z_jobs_shaheen_72hours
#!/bin/bash
#SBATCH --partition=72hours
#SBATCH --qos=72hours
#SBATCH --job-name="vasp"
#SBATCH --nodes=1
#SBATCH --extra-node-info=2:16:1
#SBATCH --time=72:00:00
#SBATCH --exclusive
#SBATCH --err=std.err
#SBATCH --output=std.out
#SBATCH --mail-type=ALL
#SBATCH --mail-user=zhiyong.zhu@kaust.edu.sa
#-------------------------------------------------------#
sleep 600
```

Figura 5.3: Sbatch + Scripting

E executalo cun só comando, obtendo como ben se dixo antes a saída típica deste:

```
[zhuz@cdl2:/scratch/tmp/zzy]$sbatch z_jobs_shaheen
Submitted batch job 1861233
```

Figura 5.4: Sbatch + Scripting (2)

Salloc

Con este comando poderemos xestionar recursos para un uso interactivo como ben dixemos antes, é dicir, sen unha execución concreta, tanto cun nó:

```
 $ salloc --nodes=1 --cpus-per-task=4 --mem=4gb --time=30:00
salloc: Pending job allocation 1408068
salloc: job 1408068 queued and waiting for resources
salloc: job 1408068 has been allocated resources
salloc: Granted job allocation 1408068
salloc: Waiting for resource configuration
salloc: Nodes ds503-01 are ready for job
```

Figura 5.5: Salloc

Coma con varios:

```
$ salloc --nodes=2 --cpus-per-task=2 --mem=4gb --time=10:00
salloc: Pending job allocation 1408075
salloc: job 1408075 queued and waiting for resources
salloc: job 1408075 has been allocated resources
salloc: Granted job allocation 1408075
salloc: Waiting for resource configuration
salloc: Nodes ds503-[13,21] are ready for job
```

Figura 5.6: Salloc (2)

Srun

Como ben comentabamos no apartado anterior, **Srun** usarémolo para correr un *job* despois de facer un *alloc* dos recursos, utilizando a *bash* devolta. Vémolo na seguinte imaxe:

```
$ srun --pty --time=1:00 --nodes=1 --ntasks-per-node=1  --mem=4g bash -l
srun: job 1408078 queued and waiting for resources
srun: job 1408078 has been allocated resources
```

Figura 5.7: Srun

Sinfo

Dentro dos comandos que catalogamos no capítulo anterior como de **monitorización**, temos en primeiro lugar **Sinfo**, o cal nos mostrará a información dos nosos nós dos *clusters* dende un punto de vista global, como podemos ver na seguintes imaxe:

```
[zhuz@cdl6:~/01]$sinfo
PARTITION AVAIL JOB_SIZE  TIMELIMIT   CPUS  S:C:T   NODES STATE      NODELIST
all       up    1-infini 1-00:00:00     64 2:16:2     16 allocated  nid000[20-27,32-39]
all       up    1-infini 1-00:00:00     20 1:10:2      8 idle       nid000[40-47]
workq*    up    1-infini 1-00:00:00     64 2:16:2     16 allocated  nid000[20-27,32-39]
knc       up    1-infini 1-00:00:00     20 1:10:2      4 idle       nid000[40-43]
kepler    up    1-infini 1-00:00:00     20 1:10:2      4 idle       nid000[44-47]
```

Figura 5.8: Sinfo

Podemos exprimir un pouco máis este comando e comezar a pasarlle diferentes filtros dunha forma moi semellante a como o facemos para as impresións en *C*, é dicir, usando porcentaxes e números, da forma que se quixeramos unha saída que nos mostrara un resumo das diferentes partes do *cluster*, co seu número de nós, o seu estado nun formato específico, como por exemplo *allocated/idle/other/total*, e algunha que outra medida, executaríamos algo do estilo:

```
$ sinfo -o "%.15P %.15s %.13a %.20l %.25F"
      PARTITION        JOB_SIZE         AVAIL           TIMELIMIT            NODES(A/I/O/T)
         batch*      1-infinite            up      14-00:00:00              336/34/6/376
          debug      1-infinite            up         2:00:00                    1/1/0/2
            amd      1-infinite            up      14-00:00:00             1/204/0/205
    group-stsda      1-infinite            up      14-00:00:00               0/38/1/39
     group-csim      1-infinite            up      14-00:00:00               0/31/0/31
     group-redd      1-infinite            up      14-00:00:00               0/11/0/11
     smc_64_128      1-infinite            up       7-00:00:00               4/80/1/85
    smc_128_512      1-infinite            up       7-00:00:00               6/80/1/87
        smc_128      1-infinite            up       7-00:00:00               4/80/1/85
```

Figura 5.9: Sinfo avanzado

Squeue

En canto outro dos comandos de monitorización temos **Squeue**, o cal nos permite ver a información, esta vez dos traballos *slurm*. É un comando moi poderoso e útil á hora de monitorizar e revisar o estado de todos os traballos do noso *cluster* e ofrece, entre outras moitas, as posibilidades e saídas que podemos ver nas seguintes imaxes:

Un listado normal dos traballos encolados:

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1324774     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
           1324773     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
           1324771     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
           1324766     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
           1324767     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
           1324768     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
           1324769     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
           1324770     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
           1324762     batch sra2bam_  kuwahah PD       0:00      1 (Priority)
```

Figura 5.10: Squeue

Comezar a sacarlle partido filtrando por exemplo por usuario e por estado **R** (*running*):

```
$ squeue -u salazaor -t R
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1205736     batch Purge_ha salazaor  R 14-04:03:36      1 ds503-13
           1219924     batch SSpLAbrS salazaor  R 10-18:38:00      1 sdlm111-22
           1219921     batch SSpLAbSA salazaor  R 10-19:23:28      1 sdlm111-18
           1205737     batch Purge_ha salazaor  R 6-20:38:16      1 dbn302-20-l
           1318898     batch  RpModel salazaor  R 1-09:30:30      1 ds506-09
           1294578     batch SSsponge salazaor  R 3-04:15:55      1 dlm104-29
           1294580     batch SSsponge salazaor  R 3-04:15:55      1 dlm104-33
```

Figura 5.11: Squeue con filtro

Ou indicándolle o formato de saída desexado:

```
$ squeue -o "%A %j %u %C %D %N %m" -u minenky
JOBID NAME USER CPUS NODES NODELIST MIN_MEMORY
1345527 m1_00_PR24_ccsdt_cc-pvtz_ecp minenky 16 1  235G
1345528 m1_00_PR25_ccsdt_cc-pvtz_ecp minenky 16 1  235G
1345526 m1_00_H2_ccsdt_cc-pvtz_ecp minenky 16 1  235G
1345588 m1_00_PR35_ccsdt_cc-pvtz_ecp minenky 16 1  500G
1345586 m1_00_PR24_ccsdt_cc-pvtz_ecp minenky 16 1  500G
1345587 m1_00_PR25_ccsdt_cc-pvtz_ecp minenky 16 1  500G
1317207 m1_00_PR09_ccsdt_cc-pvtz_ecp minenky 40 1 cn605-24-r 235G
1317210 m1_00_EDPR41_ccsdt_cc-pvtz_ecp minenky 40 1 cn603-16-l 235G
1345578 m1_00_PR07_ccsdt_cc-pvtz_ecp minenky 64 1 dlm511-01 500G
1345579 m1_00_PR08_ccsdt_cc-pvtz_ecp minenky 64 1 dlm511-13 500G
1345580 m1_00_PR09_ccsdt_cc-pvtz_ecp minenky 64 1 dlmn514-19 500G
1345582 m1_00_PR16_ccsdt_cc-pvtz_ecp minenky 64 1 dlmn514-27 500G
1345583 m1_00_PR17_ccsdt_cc-pvtz_ecp minenky 32 1 lm602-06 500G
1345584 m1_00_PR22_ccsdt_cc-pvtz_ecp minenky 32 1 lm602-08 500G
1345576 m1_00_ED25_ccsdt_cc-pvtz_ecp minenky 64 1 dlm104-25 500G
1345573 m1_00_ED07_ccsdt_cc-pvtz_ecp minenky 64 1 dlm104-37 500G
1345574 m1_00_ED08_ccsdt_cc-pvtz_ecp minenky 64 1 sdlm111-16 500G
1345575 m1_00_ED09_ccsdt_cc-pvtz_ecp minenky 64 1 dlm104-21 500G
```

Figura 5.12: Squeue con saída formatada

Sacct

E por último neste subapartado de monitorización, temos o comando de *accounting* de *slurm*. Este permitirános obter estatísticas sobre todos os datos, traballos e parte dos mesmos. Vémolo exemplificado a continuación, onde por exemplo, solicitamos primeiro unha lista das estatíticas básicas dos últimos traballos:

```
[zhuz@cdl2:/scratch/tmp/zzy]$sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
1860178           yambo      workq        k01        128     FAILED      2:0
1860178.bat+      batch                   k01         32     FAILED      2:0
1860178.0         yambo                   k01         64     FAILED      2:0
1860182           yambo      workq        k01        128 CANCELLED+      0:0
1860182.bat+      batch                   k01         32  CANCELLED     0:15
1860182.0         yambo                   k01         64  CANCELLED     0:15
1860186           yambo      workq        k01         64  COMPLETED      0:0
1860186.bat+      batch                   k01         32  COMPLETED      0:0
1860186.0         yambo                   k01         32  COMPLETED      0:0
1860190           yambo      workq        k01         64  COMPLETED      0:0
1860190.bat+      batch                   k01         32  COMPLETED      0:0
1860190.0         yambo                   k01         32  COMPLETED      0:0
```

Figura 5.13: Sacct

E no seguinte, para facelo algo máis complexo e ver as súas capacidades, solicitamos as estatísticas dos traballos dun usuario, cunha data de comezo e onde solicitamos tamén un

formato de saída personalizado:

```
$ sacct -u alonaza -S 2018-12-05 --format JobID,JobName,AllocCPUS,submit,start,end,nodelist,state,exitcode
       JobID    JobName  AllocCPUS             Submit               Start                 End        NodeList      State ExitCode
------------ ---------- ---------- ------------------- ------------------- ------------------- --------------- ---------- --------
649749         modeling         32 2018-12-04T12:11:56 2018-12-04T17:01:19 2018-12-05T06:01:21       gpu601-02    TIMEOUT      0:0
649749.batch      batch         32 2018-12-04T17:01:19 2018-12-04T17:01:19 2018-12-05T06:01:52       gpu601-02  CANCELLED     0:15
658345       modeling_+         32 2018-12-05T11:32:06 2018-12-05T11:32:37 2018-12-05T11:35:37       gpu601-02  COMPLETED      0:0
658345.batch      batch         32 2018-12-05T11:32:37 2018-12-05T11:32:37 2018-12-05T11:35:37       gpu601-02  COMPLETED      0:0
658346       modeling_+         32 2018-12-05T11:34:01 2018-12-05T11:34:56 2018-12-05T11:45:45       gpu601-05  COMPLETED      0:0
658346.batch      batch         32 2018-12-05T11:34:56 2018-12-05T11:34:56 2018-12-05T11:45:45       gpu601-05  COMPLETED      0:0
658350        rtmooc_64         32 2018-12-05T11:42:03 2018-12-05T11:44:12 2018-12-05T11:55:13       gpu601-02 OUT_OF_ME+    0:125
658350.batch      batch         32 2018-12-05T11:44:12 2018-12-05T11:44:12 2018-12-05T11:55:13       gpu601-02 OUT_OF_ME+    0:125
658675        rtmooc_64         32 2018-12-05T13:24:00 2018-12-05T13:45:30 2018-12-05T13:49:06       gpu601-08     FAILED      7:0
658675.batch      batch         32 2018-12-05T13:45:30 2018-12-05T13:45:30 2018-12-05T13:49:06       gpu601-08     FAILED      7:0
658679       rtmooc_128         32 2018-12-05T13:14:52 2018-12-05T13:16:34 2018-12-05T13:39:45       gpu601-08 OUT_OF_ME+    0:125
658679.batch      batch         32 2018-12-05T13:16:34 2018-12-05T13:16:34 2018-12-05T13:39:45       gpu601-08 OUT_OF_ME+    0:125
```

Figura 5.14: Sacct avanzado

Scontrol

Este comando *slurm* permitirános modificar *en quente* os nós e tamén os *jobs* que se estén executando, así como tamén mirar en detalle a súa información para saber qué cambiar. Polo que imos ver primeiro esta última opción, que consiste no comando de **scontrol show**, que podemos utilizalo tanto sobre nós como sobre *jobs*:

```
[zhuz@cdl2:/scratch/tmp/zzy]$scontrol show job 1861262
JobId=1861262 JobName=vasp
   UserId=zhuz(100744) GroupId=g-zhuz(1100744)
   Priority=1 Nice=0 Account=k01 QOS=normal
   JobState=PENDING Reason=Priority Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=04:00:00 TimeMin=N/A
   SubmitTime=2016-07-13T20:58:24 EligibleTime=2016-07-13T20:58:24
   StartTime=2016-07-13T23:01:42 EndTime=Unknown
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=workq AllocNode:Sid=cdl2:22290
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null) SchedNodeList=nid02070
   NumNodes=1-1 NumCPUs=32 CPUs/Task=1 ReqB:S:C:T=0:0:16:1
   TRES=cpu=32,mem=64224,node=1
   Socks/Node=2 NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=2007M MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=0 Contiguous=0 Licenses=(null) Network=(null)
   Command=/lustre/scratch/tmp/zzy/z_jobs_shaheen
   WorkDir=/lustre/scratch/tmp/zzy
   StdErr=/lustre/scratch/tmp/zzy/std.err
   StdIn=/dev/null
   StdOut=/lustre/scratch/tmp/zzy/std.out
   Power= SICP=0
```

Figura 5.15: Scontrol show sobre jobs

```
[zhuz@cdl2:/scratch/tmp/zzy]$scontrol show node nid02070
NodeName=nid02070 Arch=x86_64 CoresPerSocket=16
   CPUAlloc=64 CPUErr=0 CPUTot=64 CPULoad=32.01 Features=(null)
   Gres=craynetwork:4
   NodeAddr=nid02070 NodeHostName=nid02070 Version=15.08
   OS=Linux RealMemory=131072 AllocMem=64224 FreeMem=119024 Sockets=2 Boards=1
   State=ALLOCATED ThreadsPerCore=2 TmpDisk=0 Weight=1 Owner=N/A
   BootTime=2016-06-12T11:36:24 SlurmdStartTime=2016-06-29T13:58:24
   CapWatts=n/a
   CurrentWatts=371 LowestJoules=612028661 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

Figura 5.16: Scontrol show sobre nós

Por outro lado, como ben comentamos, podemos modificar estes mesmos da forma que se mostra na seguinte imaxe, gracias a **Scontrol**. Concretamente, cambiamos o estado dun traballo que estaba a esperar para comezar, pasándolle o *flag hold*:

```
$ sbatch --job-name=test --begin=2020-01-01T08:00:00 ./naga.1.sh
Submitted batch job 1405163
[dbn503-33-r Intel]:/home/kathirn
$ squeue -u kathirn
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1405163     batch     test  kathirn PD       0:00      1 (BeginTime)
[dbn503-33-r Intel]:/home/kathirn
$ scontrol hold 1405163
[dbn503-33-r Intel]:/home/kathirn
$ squeue -u kathirn
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1405163     batch     test  kathirn PD       0:00      1 (JobHeldUser)
```

Figura 5.17: Scontrol co flag hold

Para volver ao estado anterior, necesitariamos pasarlle esta vez un **Scontrol release** ao traballo *holdeado*. Así mesmo, poderemos suspendelo con *suspend* ou cancelalo definitivamente con *cancel*.

Scancel

Por último, teremos o comando **Scancel**, que como ben podemos deducir do seu nome, encargaráse de cancelar ou eliminar un ou varios traballos da cola. Coma o resto dos comandos, tamén aceptará *flags* para filtrar. Vemos exemplos a continuación, onde en primeiro lugar, cancelamos un traballo co seu *jobid*:

```
[zhuz@cdl2:/scratch/tmp/zzy]$squeue -u zhuz
      JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME
    1861257     zhuz     k01           vasp  PD Priority  2016-07-13T
[zhuz@cdl2:/scratch/tmp/zzy]$scancel 1861257
[zhuz@cdl2:/scratch/tmp/zzy]$squeue -u zhuz
      JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME
[zhuz@cdl2:/scratch/tmp/zzy]$
```

Figura 5.18: Scancel

E en segundo, cancelamos todos os traballos dun usuario, **-u**, esixindo unha segunda confirmación co *flag* **-i**.

```
[zhuz@cdl2:/scratch/tmp/zzy]$sbatch z_jobs_shaheen
Submitted batch job 1861259
[zhuz@cdl2:/scratch/tmp/zzy]$squeue -u zhuz
      JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME
    1861259     zhuz     k01           vasp  PD Priority  N/A
    1861258     zhuz     k01           vasp  PD Priority  2016-07-13
[zhuz@cdl2:/scratch/tmp/zzy]$scancel -u zhuz -i
Cancel job_id=1861258 name=vasp partition=workq [y/n]? y
Cancel job_id=1861259 name=vasp partition=workq [y/n]? n
[zhuz@cdl2:/scratch/tmp/zzy]$squeue -u zhuz
      JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME
    1861259     zhuz     k01           vasp  PD Priority  N/A
[zhuz@cdl2:/scratch/tmp/zzy]$
```

Figura 5.19: Scancel avanzado

Conclusións

Como vimos, aínda que con funcionalidades diferentes, os comandos de *slurm* teñen un gran parecido á hora de ser executados, xa que, a forma na que indicamos os *flags* (os cales na maioría deles tamén son idénticos, véxase o *-u* de usuario) e a maneira na que formatamos a saída desexada segue un patrón para todos eles. Queda algún que outro comando sen comentar e obviamente centos de casos de uso para cada un que probar, pero unha vez coñecemos os máis básicos e entendemos o seu funcionamento, tan só quedaría ir probando e executando a medida que nós, ou mellor dito, o noso *cluster*, o necesitara.

Por último, e antes de pasar ao seguinte apartado, comentar que este capítulo non sería posible sen os documentos sobre *HPC* publicados polo **Laboratorio de Supercomputación de Shaheen** na súa web: [3] [4] [5]

Capítulo 6

Monitorización do Cluster

Un dos aspectos máis fundamental dos *clusters* consiste en ter unha boa infraestrutura de monitorización que nos permita saber en todo momento que está ocorrendo e que ocorreu no noso *cluster*. No caso de *Qlustar*, temos varias alternativas para facelo, en concreto, por defecto xa preconfiguradas, **Ganglia e Nagios**.

Debido a que esta última require dunha maior configuración ao ter que establecer usuarios e configuración de *logueo*, ficheiros de definición de nós e *plug-ins*, etcétera, imos ver a primeira opción.

Ganglia

Por defecto, se tecleamos nun navegador web de calquera *PC* que se atope na mesma *LAN* ca o noso *Head Node*: **http://ip_headnode/ganglia**, teremos un cadro de mando ao completo de todos os nosos *clusters*. Dunha forma extremadamente sinxela como é unha petición *web* poderemos ver todas as métricas e estados do noso *cluster*.

Isto é posible xa que cada nó dos nosos *clusters* enviará información de monitorización á dirección *multicast* onde o *Head Node* a capturará e se encargará de que *Ganglia* a procese de forma axeitada para que a poidamos ver.

A continuación vemos algún exemplo das pantallas accesibles:

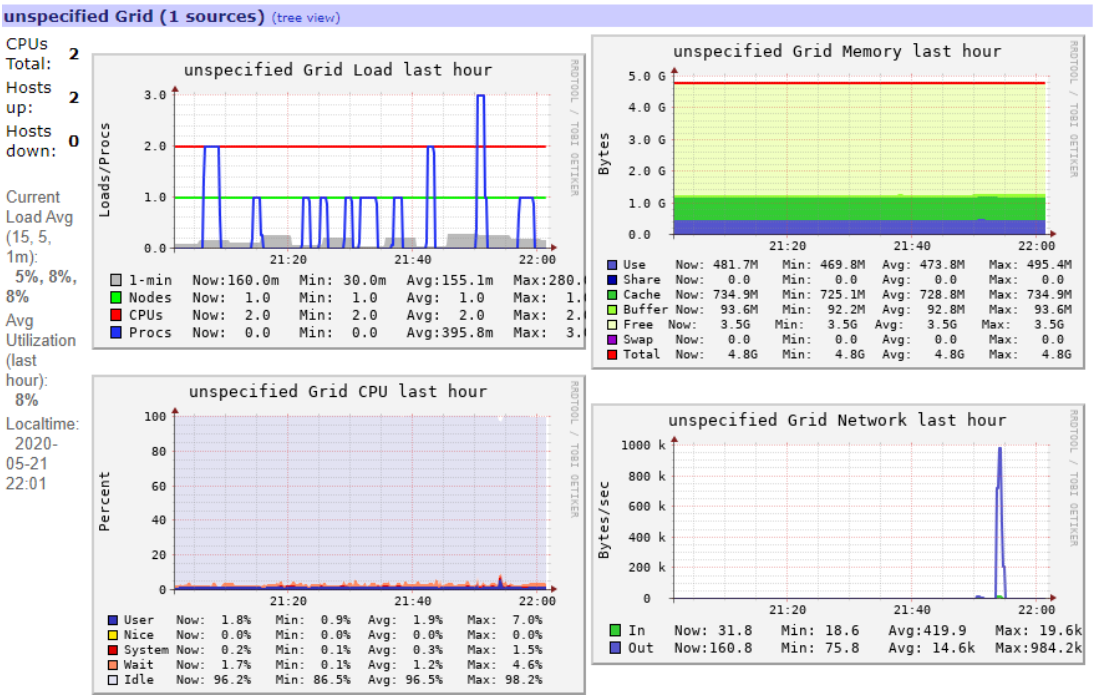

Figura 6.1: Index de Ganglia

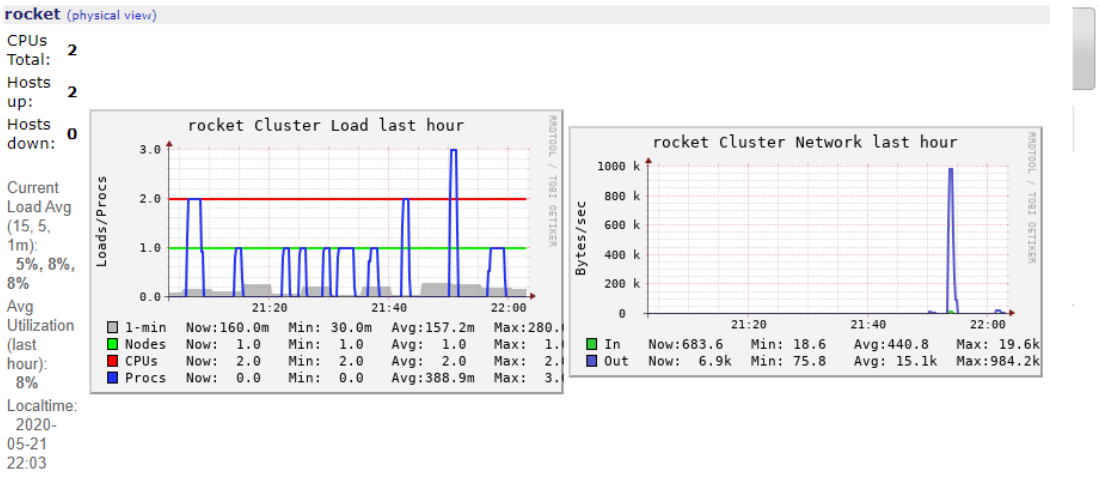

Figura 6.2: Index de Ganglia (2)

Tamén vemos na seguinte imaxe todas as opcións do menú principal que nos da *Ganglia*, onde podemos observar o potencial da ferramenta, que conta até cunha versión móbil, á parte dos múltiples e diversos filtros e métricas que aplicarlle aos *clusters*.

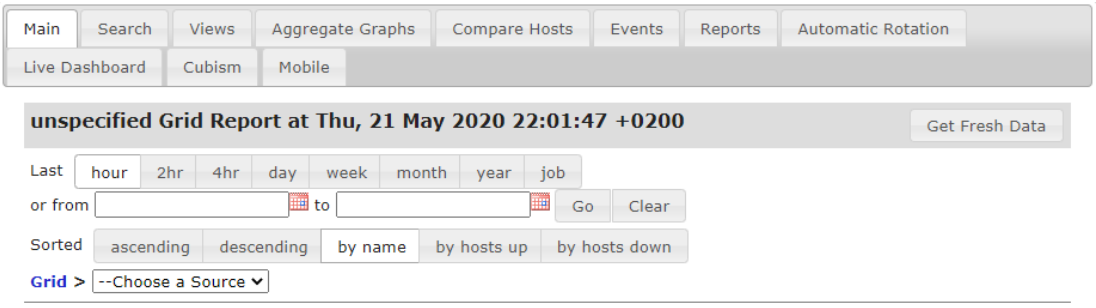

Figura 6.3: Opcións menú principal Ganglia

Como última imaxe, ver a saída que nos mostra *Ganglia* nun *cluster* real, onde vemos a actividade de centos de nós e miles de *CPUs* durante a execución de diversas tarefas [6] :

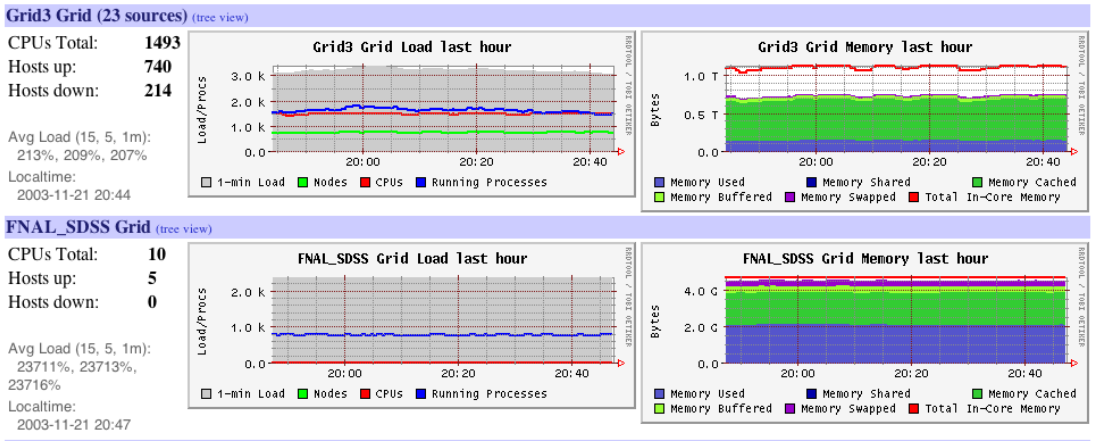

Figura 6.4: Ganglia nun cluster real

Polo que con este capítulo, sumado ao apartado de *Monitorización e Accounting* visto en *Slurm*, temos de forma resumida as principais formas de ter monitorizado ao completo o noso *cluster*, dende un punto de vista máis xeral, controlando o estado do *cluster* e dos nós, así como diversas métricas deles, ata unha aproximación moito máis concreta podendo analizar ao detalle as diferentes características de cada traballo levado a cabo no noso *cluster* cos diferentes comandos de *Slurm*.

Capítulo 7

Automatización usando Vagrant

Neste último apartado imos tratar de automatizar parte do proceso explicado con anterioridade. Concretamente, imos crear unha *Box* de **Vagrant** coa *ISO* de *Qlustar* preinstalada e preconfigurada, é dicir, automatizar o proceso que anteriormente se explicou como facer de forma manual no capítulo 1.1.

Unha vez que temos a máquina configurada como ao comezo deste capítulo, é dicir, tras todo o documento anterior, xerar unha *VagrantBox* non é complexo pero aínda así conleva unha serie de modificacións á nosa *VM* para poder facelo correctamente. Imos comentalas:

- Primeiro de todo, necesitaremos ter unha interfaz de rede tipo **NAT** activa e ademais que sexa a primeira delas. Polo que aquí vemos xa a primeira diferencia, desta vez necesitaremos **3 interfaces de rede**, as dúas comentadas na memoria máis esta extra para a nosa *Box*. A súa función será unicamente para as conexións de *Vagrant*, polo que as tarxetas de rede usadas para o *cluster* seguirán a ser as mesmas, solo que desta vez, desplazadas cara a segunda e terceira posición.

  Este requisito é o primeiro dos dous requisitos impostos por usar un *provider* específico, neste caso **VirtualBox**. O outro será instalar as **Vbox Guest Additions**, que deixaremos para o final. Antes, imos cos cambios comúns para todas as ***base-boxes***:

- Precisaremos dun **usuario *vagrant*** pertencente a un **grupo *vagrant***, polo que crearemos ambos. Aquí destacar a importancia de que o contrasinal do mesmo sexa *vagrant* e que o deberemos configurar con **ssh insecure keypair**.

- Este último será outro dos requisitos, será usado por *Vagrant* para as conexións por defecto *SSH* e para conseguilo tan só teremos que copiar a chave pública subida ao *git* oficial en [7] no ficheiro **/home/vagrant/.ssh/authorized_keys**. Tras facer isto, deberemos asegurarnos que este directorio ten permisos **0700** e que o ficheiro ca chave ten permisos **0600**.

- Por outro lado, é recomendable que o contrasinal de **root** sexa tamén **vagrant**, esta opción, como xa vimos na memoria, elexiráse na primeira configuración de *Qlustar* e aínda que non sexa fundamental, é unha boa práctica.
- Un axuste crítico será facer ao **usuario vagrant password-less sudo**, para isto, procederemos a editar o ficheiro *sudoers*, nunca de forma manual, sempre co comando ***visudo*** e engadindo ao final unha liña como:

```
vagrant ALL=(ALL) NOPASSWD: ALL
```

  A cal dotará ao usuario *vagrant* dos permisos para poder executar todos os comandos con *sudo* sen necesidade de introducir ningún contrasinal, o que facilitará enormemente o proceso de automatización.
- Por último, e como segunda medida non fundamental pero si recomendada, deberemos deshabilitar o *flag* ***UseDNS*** na configuración *ssh* (**/etc/ssh/sshd_config**) para optimizar as conexións aforrando segundos de *reverse DNS lookups*.
- Unha vez que temos todos os pasos comúns completados, procederemos finalmente a instalar as **VBox Guest Additions**. Isto será un proceso idéntico á instalación das mesmas en calquera distro *Linux*, polo que como xa é unha actividade coñecida, non a imos comentar en profundidade. Simplemente despois de instalar as **linux-headers** correspondentes e os paquetes **build-essential** e **dkms**, executaremos o *script* no *CD* das *guests*, o cal instalará correctamente as mesmas e tras un ***reboot***, quedarán completamente operativas.

Polo que chegados até aquí, contaremos ca nosa *VM* con todo o *SW* necesario para crear unha ***Vagrant Basebox*** a partir dela, agora tan só quedaría instalar o *SW* extra necesario que consideremos e facer as configuracións desexadas para a nosa *box* a exportar. Unha vez feito, tan só executariamos o comando de *Vagrant*:

```
vagrant package --base my-virtual-machine
```

O cal, tras un período de tempo, nos daría a nosa desexada *box*. Se queremos probar que todo foi ben, podemos tratar de importala e desplegar unha nova máquina clon a partir dela cos comandos:

```
$ vagrant box add --name my-box /path/to/the/new.box
$ vagrant init my-box
$ vagrant up
```

Tras isto, a *MV* sería despregada correctamente e teriamola lista para comezar a explotar *Qlustar*. Gracias a este *box*, dotamos dun alto nivel de automatización á configuración, así como dunha gran portabilidade ao entorno. Podemos descargalo para as nosas probas en [8].

Apéndices

Apéndice A

Erros coñecidos

A continuación expóñense os erros e as solucións coñecidas que se foron presentando durante a realización deste traballo tutelado. Tamén figuran problemas actuais que o seu amaño podería ser considerado como parte dun capítulo de *Next Steps*:

- Problema co **DNS** nada máis iniciar a máquina. Se eleximos **DHCP** como configuración da interfaz de rede externa do noso *cluster*, atoparémonos con que somos incapaces de resolver nomes de dominio. Isto pódese solucionar de dúas maneiras, ambas comprobadas:
  - Configurando **estáticamente** a rede. Ca desventaxa da escalibilidade ao estar condicionando os rangos da rede con conexión a Internet na que vai correr o *cluster*.
  - Facendo un *start* e un *enable* do servizo **systemd-resolved.service**, pero ollo, perderemos os nomes de hosts por defecto que nos poden interesar (como o do noso nó *FE* ou os virtuais), polo que habería que traballar a partir deste cambio cas súas *IP's* directamente, introducindo as entradas en */etc/hosts* ou volvendo desactivar o servizo.
- Erro ao xerar o ***token* para *Qluman***, quédase parada a terminal durante moito tempo para despois emitir un erro de que non existe un usuario *admin*. Este erro está relacionado co *DNS*, xa que ocorre cando levantamos o servizo anterior. Para solucionalo, simplemente deberiamos engadir ao ficheiro **/etc/hosts** unha entrada co nome e ca *IP* externa do noso *Head*, é dicir: **IP_headnode - beosrv-c**.
- Non se levantan correctamente os nós virtuais do cluster, no noso caso o **FE VM** e os **dous nós demo virtuais** de cómputo modo demo. A día de hoxe, e despois de recibir resposta no foro oficial de *Qlustar*, parece practicamente imposible solucionar isto xa que o problema recae en que *Qlustar* non soporta que o noso *Head Node* sexa unha

MV de *VirtualBox*, polo que para arreglalo habería que practicamente modificar metade do traballo, cousa inviable. Aínda así, queda referenciada a continuación a entrada do foro onde suxiren posibles solucións ao problema [16] e se se quixera solucionar de raíz simplemente poderiamos optar por facer todos os pasos da memoria sobre unha máquina física e non virtual, tendo así o noso *Head Node* nun *PC* dedicado, por exemplo.

- Problemas co **SSH** na nosa *Vagrant Base Box*. Aínda que o problema está acotado, xa que o erro ocorre debido á incorrecta conexión *SSH* en modo *insecure key pair*, as solucións atopadas son *workarounds* e non unha solución de raíz. Éstas son:

  - O resto do proceso de arranque da *VM* da nosa *Base Box* é correcto polo que para solucionalo poderiamos acceder á **GUI de VirtualBox**, mostrar a pantalla da máquina despregada con *Vagrant Up*, e loguearnos cas credenciais **vagrant - vagrant**
  - Outra opción sería cambiar a configuración *SSH* en **/etc/ssh/sshd_config**, para engadir ao usuario *vagrant* aos **AllowedUser**, e unha vez feito o *vagrant up*, dende outra máquina, facer un **ssh vagrant@IP_VM**, o que nos proporcionaría tamén acceso á nosa *VM*.

  Aínda despois de varias comprobacións e probas, cambiando a chave pública almacenada en **authorized_keys**, probando a facer un **ssh -I private_key** indicando explícitamente a chave privada do noso *insecure key pair* e dando os permisos e seguindo as instruccións indicadas na documentación de *Vagrant* [15], non se logrou solucionar este erro, en gran medida debido á pouca experiencia á hora de crear *Vagrant Base Boxes* propias, polo que aínda que solucionado mediante *workarounds* queda como materia pendente unha solución da causa do problema.

- Algunhas veces, ocorre que despois de facer o *vagrant up*, as interfaces de rede son incapaces de levantarse. A solución á que cheguei, quizais non a máis efectiva pero si a máis sinxela, é que dende a **GUI de Virtualbox**, con marcar a casilla de activación da interfaz/interfaces desexadas, soluciónase. Hai que ter ollo xa que é un erro que non ocorre sempre pero que pode levar a tempos de espera longos se non o identificamos rapidamente.

Bibliografía

[1] "Manual de instalación de qlustar os," https://docs.qlustar.com/en-US/Qlustar_Cluster_OS/11.0/html-single/Installation_Guide/index.html, accedido en maio de 2020.

[2] "Descarga da imaxe de qlustar," https://qlustar.com/download, accedido en maio de 2020.

[3] "Monitoring and managing ibex jobs," https://www.hpc.kaust.edu.sa/sites/default/files/files/public/Cluster_training/21_02_2019/SLURM_job_management.pdf, accedido en maio de 2020.

[4] "Application examples," https://www.hpc.kaust.edu.sa/sites/default/files/files/public/KSL/150520-User_Workshop/KSL_WS_Zhiyong.pdf, accedido en maio de 2020.

[5] "Some useful slurm commands," https://www.hpc.kaust.edu.sa/sites/default/files/files/public/20160710%20SlurmCommands.pdf, accedido en maio de 2020.

[6] "Ganglia," http://sistemas-distribuidos-paralelos.blogspot.com/2012/05/ganglia.html, accedido en maio de 2020.

[7] "Chave pública do insecure pair de vagrant," https://raw.githubusercontent.com/hashicorp/vagrant/master/keys/vagrant.pub, accedido en maio de 2020.

[8] "Vagrant base box resultante da memoria," https://udcgal-my.sharepoint.com/:u:/g/personal/mauro_delossantos_udc_es/EX0WQ5AclURNt_aDXmZ63ZoB9khwcQlzyvabiqQ0ZbawfA, accedido en maio de 2020.

[9] "Primeiros pasos con qlustar," https://docs.qlustar.com/en-US/Qlustar_Cluster_OS/11.0/html-single/First_Steps_Guide/index.html, accedido en maio de 2020.

[10] "Guía de qluman," https://docs.qlustar.com/en-US/Qlustar_Cluster_OS/11.0/html-single/QluMan_Guide/index.html#chap-Adding-Hosts, accedido en maio de 2020.

[11] "Guía xeral de administración dos clusters," https://docs.qlustar.com/en-US/Qlustar_Cluster_OS/11.0/html-single/Administration_Manual/index.html#admin-man-sect-nagios, accedido en maio de 2020.

[12] "Qlustar hpc stack," https://docs.qlustar.com/en-US/Qlustar_HPC_Stack/11.0/html-single/HPC_User_Manual/index.html#chap-hpc-user-man-shell-setup, accedido en maio de 2020.

[13] "Cluster handbooks," https://en.wikibooks.org/wiki/Cluster-Handbook, accedido en maio de 2020.

[14] "Creating a base box," https://www.vagrantup.com/docs/boxes/base.html, accedido en maio de 2020.

[15] "Creating a base box with virtualbox," https://www.vagrantup.com/docs/virtualbox/boxes.html, accedido en maio de 2020.

[16] "Qlustar mailing list," https://lists.qlustar.org/hyperkitty/list/qlustar-general@qlustar.org/thread/KVXRXGNJPVM7NI6R4Q4E5HQGWHIIJ6ZG/, accedido en maio de 2020.