Slurm Roll for Rocks Cluster

Werner Saar

November 25, 2015

Contents

Chapter 1

Introduction

The Slurm Roll provides a resource manager and scheduler for the Rocks Cluster System. Slurm cannot coexist with other batch systems like Torque or Sun Grid Engine.

Summary

Name	slurm
Version	6.2
Maintained by	Werner Saar
Architecture	x86_64
Compatible with Rocks®	6.2

Roll Compatibility

The Slurm Roll has been tested with the following rolls.

```
NAME          VERSION   ARCH        ENABLED REQUIRED
base:         6.2       x86_64      yes       yes
ganglia:      6.2       x86_64      yes       yes
hpc:          6.2       x86_64      yes       yes
kernel:       6.2       x86_64      yes       yes
os:           6.2       x86_64      yes       yes
web-server:   6.2       x86_64      yes       yes
java:         6.2       x86_64      yes       no
perl:         6.2       x86_64      yes       no
python:       6.2       x86_64      yes       no
```

Included Software on x86_64

slurm	https://computing.llnl.gov/linux/slurm/
hwloc	http://www.open-mpi.org/projects/hwloc/
munge	http://code.google.com/p/munge/
python-hostlist	http://www.nsc.liu.se/ kent/python-hostlist/
freeipmi	http://www.gnu.org/software/freeipmi/

Support

The Slurm Roll is provided AS-IS without support. Please send email to wernsaar@googlemail.com for free support.

License

Installation

The Slurm Roll is provided as iso image. The normal way to install is:

```
export LANG=C
rocks add roll slurm*.iso
rocks enable roll slurm
cd /export/rocks/install
rocks create distro
yum clean all
yum update
rocks run roll slurm|sh
reboot
```

Update

To prevent, that the files partitions.conf, topology.conf and nodenames.conf are created/overwritten, you should add the rocks attribute slurm_sync_enable with the value false. The value false is case sensitiv.

Example:

rocks add attr slurm_sync_enable false

```
export LANG=C
rocks disable roll slurm
rocks remove roll slurm
rocks add roll slurm*.iso
rocks enable roll slurm
cd /export/rocks/install
rocks create distro
yum clean all
yum update
service slurmdbd restart
service slurm restart
```

Now You can reinstall your compute nodes. The slurm roll uses the default mysql server. Update to slurm-14.11.x and later from versions < 2.5.0 is not supported.

Chapter 2

Admin Guide

In it's default configuration slurm is set up for fairshare scheduling. There are seven configuration files in the directory /etc/slurm.

1. slurm.conf
2. topology.conf
3. slurmdbd.conf
4. nodenames.conf
5. partitions.conf
6. headnode.conf
7. cgroup.conf

All these files must be replicated to the compute nodes. Actually this is done by 411 (see /var/411/Files.mk). There is also as special command **rocks sync slurm** to replicate the configuration files. As soon as Slurm provides a better way, this will be changed. For better support of heterogeneous clusters the features of compute nodes are now automatically generated. Additional features can be added with the attribute slurm_feature. With the attribute slurm_properties it is possible to flag compute nodes to have some special properties.

With the host attribute **slurm_properties** it is possible to override properties of a compute node.

Example:

rocks add host attr compute-1-0 slurm_properties value='CoresPerSocket=2'

Some features will be set automatically, but it's also possible to add some features with the host attribute **slurm_feature**

Example:

rocks add host attr compute-1-0 slurm_feature value='2GPUs'

Rocks commands

There are some slurm roll related commands.

rocks report slurmnodes

rocks report slurmpartitions

rocks report slurm_memory
The output of this command can piped into a shell to set the host attributes slurm_memory and slurm_cpuspeed

rocks report slurm_hwinfo
The output of this command can be piped into a shell to set the host attribute slurm_hwinfo. Some values from the command slurmd -C are used, to set the cpu layout.

The reports above are used to generate the configuration files in /etc/slurm.
rocks sync slurm generates and replicates the configuration files.

Slurm can be configured to control interactive access to compute nodes. When enabled a user can only login to nodes, where he has running jobs or has allocated some resources. This behavior can be changed by setting the global or host attribute slurm_pam_enable to true or false. By default the attribute slurm_pam_enable is set to true. You have to run rocks sync slurm or reboot the compute nodes after changing the attribute slurm_pam_enable. Root and members of the group wheel are not affected by this flag.

X11 Applications

If users want to run X11 applications on compute nodes, the following entry should be added to /etc/ssh/sshd_config:

X11UseLocalhost no

Restart sshd with the command: service sshd restart

Additional login nodes

Additional login nodes must manually configured in the file /etc/slurm/slurm.conf.

Example:

PartitionName=DEFAULT AllocNodes=rocks-61,login-2-0 State=UP

Partitions

The partition CLUSTER is always created and contains all compute nodes. If you want to add additional options for this partition, then create the file /etc/slurm/CLUSTER.options and add all options in the first line.

Example:

MaxNodes=10 GraceTime=3600

If you neeed additional partitions, then create the file /etc/slurm/partitions with one partition definition per line.

Example:

PartitionName=WHEEL AllowGroups=wheel,root

PartitionName=TEST AllowAccounts=root

The assignment of compute nodes to partitions is done with the host attribute slurm_partitions.

Examples:

rocks add host attr compute-0-0 slurm_partitions value='|WHEEL|'

rocks add host attr compute-0-1 slurm_partitions value='|WHEEL|TEST|'

Node Weight

It's now possible to modify the algorithm, how the weight of compute nodes are computed. You can define the rocks attr slurm_weight.

Example:

rocks add appliance attr compute slurm_weight value='mem*10+cpus*1000 -(rack+1)*100 - rank'

rocks sync slurm

Topology

It’s now possible to generate a network topology file. This may be useful, if you have for example an Infiniband, Myrinet or 10G Ethernet fabric. Please look at the file

/opt/rocks/lib/python2.6/site-packages/rocks/commands/report/slurm_topology/___init___.py

how to customize the topology for your network or simply send a mail to wernsaar@googlemail.com for free support.

By default, one leaf switch per rack is created and all switches are connected to one core switch. You can view the result with the command:

rocks report slurm_topology

To enable the topology, you have to do the following steps:

1. touch the file /etc/slurm/sync_topoplogy
2. remove the line NodeName=<your headnode> ... from /etc/slurm/slurm.conf
3. set TopologyPlugin=topology/tree in /etc/slurm/slurm.conf
4. execute rocks sync slurm
5. execute scontrol show topology

Chapter 3

Users Guide

Normally a user is logged in on the head-node and uses slurm commands to run interactive jobs or batch jobs on the compute nodes. The slurm scheduler decides which compute nodes to use. When starting a job, you should give some informations such as number of nodes/cpus, amount of memory, max. runtime etc. to the scheduler. There are also some commands to watch and manage the jobs. See the Slurm Quickstart Guide at https://computing.llnl.gov/linux/slurm/quickstart.html

Interactive Jobs

A great feature of Slurm is the allocation of resources for interactive jobs. This has to be done with the command **salloc**. First login on the Head-Node with the command ssh -Y ... and type for example: **salloc –exclusive -N1** to allocate a host. You can verify the allocation by typing **srun hostname**

Examples for direct login:

salloc –exclusive -n1 srun -n1 -N1 –pty –preserve-env $SHELL
salloc –exclusive -n1 srun -n1 -N1 –pty $SHELL
salloc -t 0 –exclusive -n4 -N1 ssh -Y 'srun hostname'

Batch Jobs

Suchs jobs are running in the background. You should write a short script, that contains all informations and commands to run. A batch job is then started with the command **sbatch <name of your script>**. Here is an example script:

```
#!/bin/bash
#SBATCH -J test-1-3-6
#SBATCH --partition CLUSTER
#SBATCH --nodes=1-2
#SBATCH --ntasks=6
#SBATCH --mem-per-cpu=150
#SBATCH --time=02:00:00

source /etc/profile
srun <your command>
```

The name of the job is test-1-3-6. Six tasks are needed on one or two nodes. Every task needs 150 MB memory. The estimated runtime is about two hours. By default both standard output and standard error are directed to a file of the name "slurm-%j.out", where the "%j" is replaced with the job allocation number. For running MPI jobs, You only have to use mpirun instead of srun. This will work with openmpi, mpich2 and mvapich2.

X11 applications

If You want to run x11 applications on compute nodes, first login to the head node or a login node with the command:

ssh -Y ....

Now check your DISPLAY variable with the command:

```
echo $DISPLAY
```

The output should not point to localhost, but to the name of the host, where You are logged in. Now alloc some resources with the command salloc and try to run an application.

Examples:

```
    srun xclock
    mpirun xclock
```

Chapter 4

GPU Computing

The slurm-roll now supports the integration of compute nodes, that have one or more gpu devices. The slurm roll does not provide scripts or rolls, to install closed-source gpu drivers or libraries for some reasons:

1. There may be license restrictions
2. The driver version must match the hardware

After a fresh install on a new head-node, the cluster configuration is prepared for gpu computing. If you update the slurm-roll from previous versions, you have to create and modify a few files.

Templates

A fresh install provides four templates with the names gres.conf.1 - gres.conf.4 and configures 411, to sync the templates to the compute nodes. If you only updated slurm and you want to use gpu's, you have to create a least one template and you have to configure 411.

Examples:

```
# Example for two Nvidia GPU's
Name=gpu Type=nvidia File=/dev/nvidia0 CPUs=0
Name=gpu Type=nvidia File=/dev/nvidia1 CPUs=1
```

```
# Example for two AMD/ATI GPU's
Name=gpu Type=amd File=/dev/ati/card0 CPUs=0
Name=gpu Type=amd File=/dev/ati/card1 CPUs=1
```

```
# Example for Linux framebuffer
Name=gpu Type=xxx File=/dev/fb0 CPUs=0
```

```
# Example for Testing without a real gpu
Name=gpu File=/dev/loop0 CPUs=0
Name=gpu File=/dev/loop1 CPUs=0
Name=gpu File=/dev/loop2 CPUs=1
Name=gpu File=/dev/loop3 CPUs=1
```

411 Registration

After a fresh install of the slurm-roll, the tail of the file /var/411/Files.mk looks like this:

```
FILES += /etc/slurm/slurm.conf
FILES += /etc/slurm/headnode.conf
FILES += /etc/slurm/nodenames.conf
FILES += /etc/slurm/partitions.conf
FILES += /etc/slurm/topology.conf
FILES += /etc/slurm/cgroup.conf
FILES += /etc/slurm/gres.conf.1
FILES += /etc/slurm/gres.conf.2
FILES += /etc/slurm/gres.conf.3
FILES += /etc/slurm/gres.conf.4
FILES_NOCOMMENT += /etc/munge/munge.key
```

If it was only an update, there may be no lines for the gres templates. Add the lines for the templates, that you have created. After modifying Files.mk, you have to execute three commands:

```
cd /var/411
make clean
make
```

Configuration of Compute Nodes

You only have to define the host attributes **slurm__gres** and **slurm__gres__template**.

Example:

```
rocks set host attr compute-0-1 slurm__gres__template value="gres.conf.3"
rocks set host attr compute-0-1 slurm__gres value="gpu"
```

Now you can run the command **rocks sync slurm**. After a few seconds, you can run the command **scontrol show node** and you should see an output like this:

```
NodeName=compute-0-1 Arch=x86_64 CoresPerSocket=1
CPUAlloc=0 CPUErr=0 CPUTot=1 CPULoad=0.00 Features=rack-0,1CPUs
Gres=gpu:2
NodeAddr=10.1.255.252 NodeHostName=compute-0-1 Version=14.11
OS=Linux RealMemory=2006 AllocMem=0 Sockets=1 Boards=1
State=IDLE ThreadsPerCore=1 TmpDisk=18699 Weight=20481100
BootTime=2015-03-25T17:25:55 SlurmdStartTime=2015-03-25T18:57:00
CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

If all compute nodes have the same type and number of gpus, you should not use host attributes but appliance attributes.

Example:

```
rocks set appliance attr compute slurm__gres__template value="gres.conf.3"
rocks set appliance attr compute slurm__gres value="gpu"
```

GPU Appliance

If only some compute nodes have the same type and number of gpus, it's better to define a new appliance. The following example shows step by step, how to define a new appliance called compute-gpu.

1. Graph File

Create a new graph file compute-gpu.xml in the directory /export/rocks/install/site-profiles/6.1.1/graphs/default.

Example:

```
<?xml version="1.0" standalone="no"?>

<graph>

<description>
</description>

<changelog>
</changelog>

<edge from="compute-gpu">
      <to>compute</to>
</edge>

<order gen="kgen" head="TAIL">
      <tail>compute-gpu</tail>
</order>

</graph>
```

2. Node File

Create a new node file compute-gpu.xml in the directory /export/rocks/install/site-profiles/6.1.1/nodes.

Example:

```
<?xml version="1.0" standalone="no"?>

<kickstart>

<description>
compute-gpu
</description>


<changelog>
</changelog>

<post>

<file name="/etc/motd" mode="append">
Compute GPU Appliance
</file>

</post>

</kickstart>
```

3. Create the Appliance

Execute the command:
rocks add appliance compute-gpu membership="Compute-gpu" node="compute-gpu"

The appliance and node names must begin with the string compute. The membership name must begin with the string Compute. All names are case sensitiv.

4. Define Appliance Attributes

Example:

```
rocks set appliance attr compute-gpu slurm__gres__template value="gres.conf.3"
rocks set appliance attr compute-gpu slurm__gres value="gpu"
```

If you want to start X11 on the compute-gpu nodes, then set the attribute x11.

```
rocks set appliance attr compute-gpu x11 true
```

5. Create Distro

Run the following commands:

```
cd /etc/rocks/install
rocks create distro
```

6. Replace Compute Nodes

To replace a compute node, power down (not shutdown) the node, run insert-ethers with the flag –replace, choose Compute-gpu, and power up the node

Example:

```
insert-ethers –replace compute-0-0
```

Chapter 5

Green Computing

High energy consumption is a major cost factor and bad for the environment. Slurm provides a mechanism, called Power Saving, to automatically suspend(shutdown) compute nodes, when they are idle and resume(boot) nodes, when they are needed. To shutdown a compute node is a simple task, but to resume(boot) a node, special hardware is needed. The slurm-roll tries to minimize your efforts to achieve this goal, but there still remain some tasks, you have to do.

1. Task - Check the hardware

Check the BIOS of the compute nodes, wether Wake-On-Lan is supported and enabled. If the compute nodes support Wake-On-Lan, you can use this method out-of-the-box. To check for Wake-on-Lan, execute for example the following command as user root on all compute nodes:

ethtool eth0|grep Wake

If the output looks like:

```
Supports Wake-on: pumbg
Wake-on: g
```

Wake-On-Lan is supported, correctly configured and ready for usage.

If Wake-On-Lan is not supported, check your hardware for integrated management like IPMI, ILO, DRAC ... or you can use an external power switch. The slurm-roll provides the rpm package fence-agents from the CentOS distribution, to deal with integrated management and external power switch solutions.

2. Task - Configure the hardware

If Wake-On-Lan is supported, but Wake-on is not set to g, you can try to enable this mode:

- Login as root on a compute node
- Execute for example: ethtool -s eth0 wol g

If you want or need to use an integrated management or an external power switch solution, please read the vendor documentation.

3. Task - Test to boot a node

Shutdown a compute node for example compute-0-0.
Using Wake-On-Lan is quite simple:

```
L_INTERFACE=$( /opt/rocks/bin/rocks list host interface localhost|awk ' /private/ { print $2 } ' )
R_MAC=$( /opt/rocks/bin/rocks list host interface compute-0-0|awk ' /private/ { print $3 } ' )
ether-wake -i $L_INTERFACE $R_MAC
```

Without Wake-On-lan, you should install the package fence-agents on the head.node:

yum install fence-agents

You will find a lot of programs with the command ls /usr/sbin/fence* and corresponding manual pages. Example:

```
# Name of the KVM host
R_HOST=192.168.202.2
# Name of the virtual machine
R_NODE=comp-0-0
fence_virsh --retry-on=3 --ip=$R_HOST --ssh --action=on -l root -k /root/.ssh/id_rsa  -n $R_NODE
```

Do not continue until booting of compute nodes succeds.

4. Task - Create or modify required scripts

The script /etc/slurm/hibernate.sh must exist on all compute nodes:

```
#!/bin/bash
ACTION=halt
if [ $# -gt 0 ]
then
  ACTION=$1
fi

case $ACTION in
hibernate)
   (sleep 10 && pm-hibernate ) > /dev/null 2>&1 &
   ;;
suspend)
   (sleep 10 && pm-suspend ) > /dev/null 2>&1 &
   ;;
halt)
   (sleep 10 && halt ) > /dev/null 2>&1 &
   ;;

*)
   logger -p local3.info "ACTION $ACTION not implemented"
   ;;
esac
exit 0
```

The script /etc/slurm/suspendhost.sh is quite simple and should work:

```
#!/bin/bash
logger -p local3.info "slurm suspend request $@"
if [ $# -gt 0 ]
then
for h in $( scontrol show hostnames $1 )
do
   MEMBER=$(/opt/rocks/bin/rocks report host attr $h attr=membership)
   if [[ "${MEMBER,,}" = "compute"* ]]
   then
     ACTION=$( /opt/rocks/bin/rocks report host attr $h attr=slurm_suspend_method )
     RET=$?
     if [ $RET -ne 0 ]; then
        ACTION=halt
     fi

     logger -p local3.info "suspend host $h"
     ssh $h /etc/slurm/hibernate.sh $ACTION

   else
      logger -p local3.info "not suspending host $h"
   fi
done
fi
```

The default suspend action is halt, but you can use another action(hibernate or suspend) with the attribute slurm__suspend__method.

The script /etc/slurm/resumehost.sh in only an example. It should work for Wake-On-Lan, but you have to modify this script for other methods.

```
#!/bin/bash
logger -p local3.info "slurm resume request $@"
if [ $# -gt 0 ]
then
for h in $( scontrol show hostnames $1 )
do

    R_METHOD=$( /opt/rocks/bin/rocks report host attr $h attr=slurm_resume_method )
    RET=$?
    if [ $RET -ne 0 ]; then
       R_METHOD=wol
    fi

    R_HOST=$( /opt/rocks/bin/rocks report host attr $h attr=slurm_resume_host )
    RET=$?
    if [ $RET -ne 0 ]; then
       R_HOST=localhost
    fi

    R_NODE=$( /opt/rocks/bin/rocks report host attr $h attr=slurm_resume_nodeid )
    RET=$?
    if [ $RET -ne 0 ]; then
       R_NODE=$h
    fi

    logger -p local3.info "trying to resume host $R_NODE with method $R_METHOD via $R_HOST ..."

    if [ $R_METHOD == "wol" ]; then

        L_INTERFACE=$( /opt/rocks/bin/rocks list host interface localhost|awk ' /private/ { print
        R_MAC=$( /opt/rocks/bin/rocks list host interface $h|awk ' /private/ { print $3 } ' )
        logger -p local3.info "running ether-wake -i $L_INTERFACE $R_MAC"
        ether-wake -i $L_INTERFACE $R_MAC
    fi

    case $R_METHOD in

    fence_virsh)
       logger -p local3.info "running fence_virsh --retry-on=3 --ip=$R_HOST --ssh --action=on -l
       fence_virsh --retry-on=3 --ip=$R_HOST --ssh --action=on -l root -k /root/.ssh/id_rsa  -n $
       ;;


    *)
       logger -p local3.info "resume method $R_METHOD not implemented"
       ;;

    esac

    done
    fi
```

The script resumehost.sh uses the attributes slurm_resume_method, slurm_resume_host and slurm_resume_nodeid.

Modify the script resumehost.sh for your hardware, set the attributes and check that suspendhost.sh and resumehost.sh are working correct.

5. Task - Measure boot and shutdown time

You should have a console to a compute node.

Shutdown or suspend the node and measure the time until the node is really down.
Now boot the node and measure the time until the node is really up.

You can now compute some important parameters:

- SuspendTimeout > time for suspend or shutdown
- ResumeTimeout > time for resume or boot
- SuspendTime > SuspendTimeout + ResumeTimeout

  Example:

  time for suspend or shutdown = 30 seconds
  time for resume or boot = 300 seconds

  SuspendTimeout = 45
  ResumeTimout = 450
  SuspendTime = 600

6. Task - Configure Slurm

Insert this section into the file /etc/slurm/slurm.conf, if the sections does not exist or edit the section. Use the values, that you have computed in the 5. task. Exclude some nodes from power save, at least the headnode should be in this list. Disable power save by setting SuspendTime=-1.

```
####### Power Save Begin ##################
SuspendExcNodes=<your headnode>,compute-1-0
SuspendProgram=/etc/slurm/suspendhost.sh
SuspendRate=4
# SuspendTime = 600
SuspendTime=-1
SuspendTimeout=45
ResumeProgram=/etc/slurm/resumehost.sh
ResumeRate=4
ResumeTimeout= 450
####### Power Save End   ##################
```

Execute: rocks sync slurm

7. Task - Inform and instruct users

Inform your users that you want to activate Power Save and explain the reason.
Instruct your users that it can sometimes take a little bit longer until all requested compute nodes are available and that the first statement in a batch script or after salloc should be:

scontrol wait_job $SLURM_JOBID

or that they should use the flag:

–wait-all-nodes=1

Srun will always wait until the node is ready.

8. Task - Test suspend and resume

You should have two terminal sessions to the headnode. In the first terminal type:

tail -f /var/log/messages /var/log/slurm/slurmctld.log

Run for example the following test in the second terminal:

```
salloc -N1 -w compute-0-0 --exclusive
srun hostname
/etc/slurm/suspendhost.sh compute-0-0
# look at terminal 1: wait until the node is down
/etc/slurm/resumehost.sh compute-0-0
# look at terminal 1: wait until the node is up
srun hostname
```

9. Task - Enable Power Save and test

You should have two terminal sessions to the headnode. In the first terminal type:

tail -f /var/log/messages /var/log/slurm/slurmctld.log

To enable Power Save, edit /etc/slurm/slurm.conf and set SuspendTime to the value, that you computed in the 5. task.
Execute: rocks sync slurm

Look at terminal 1: the compute nodes will be suspended.

Run now for example the following test in the second terminal:

```
salloc -N1 -w compute-0-0 --exclusive
time scontrol wait_job $SLURM_JOBID
srun hostname
```