serverfault

Howto set up SGE for CUDA devices?

Asked 13 years, 1 month ago Modified 11 years, 6 months ago Viewed 12k times

5

I'm currently facing the problem of integrating GPU-Servers into an existing SGE environment. Using google I found some examples of Clusters where this has been set up but no information on how this had been done.

Is there some form of howto or tutorial on this anywhere? It doesn't have to be ultra verbose but it should contain enough information to get a "cuda queue" up and running...

Thanks in advance...

Edit: To set up a load sensor about how many GPUs in a node are free, I've done the following:

- set the compute mode of the GPUs to exclusive
- set the GPUs to persistent mode
- add the following script to the cluster configuration as load sensor (and set it so 1 sec.)

```
#!/bin/sh

hostname=`uname -n`

while [ 1 ]; do
  read input
  result=$?
  if [ $result != 0 ]; then
    exit 1
  fi
  if [ "$input" == "quit" ]; then
    exit 0
  fi


  smitool=`which nvidia-smi`
  result=$?
  if [ $result != 0 ]; then
    gpusav=0
    gpus=0
  else
    gpustotal=`nvidia-smi -L|wc -l`
    gpusused=`nvidia-smi |grep "Process name" -A 6|grep -v +-|grep -v \|=|grep -v
Usage|grep -v "No running"|wc -l`
    gpusavail=`echo $gpustotal-$gpusused|bc`
  fi

  echo begin
  echo "$hostname:gpu:$gpusavail"
  echo end
done
```

```
exit 0
```

Note: This obviously works only for NVIDIA GPUs

cuda gridengine

Share Improve this question Follow

edited Oct 18, 2011 at 9:08

asked Oct 17, 2011 at 8:24

luxifer
177 1 3 12

4 Answers

Sorted by: Highest score (default)

5

The strategy is actually fairly simple.

Using `qconf -mc` you can create a complex resource called `gpu` (or whatever you wish to name it). The resource definition should look something like:

```
#name               shortcut   type        relop   requestable consumable default
urgency
#-------------------------------------------------------------------------------------
----------
gpu                 gpu        INT         <=      YES         YES         0        0
```

Then you should edit your exec host definitions with `qconf -me` to set the number of GPUs on exec hosts that have them:

```
hostname              node001
load_scaling          NONE
complex_values        gpu=2
user_lists            NONE
xuser_lists           NONE
projects              NONE
xprojects             NONE
usage_scaling         NONE
report_variables      NONE
```

Now that you've set up your exec hosts, you can request gpu resources when submitting jobs. eg: `qsub -l gpu=1` and gridengine will keep track of how many GPUs are available.

If you have more than one job running per node that uses a GPU you may want to place your GPUs in to exclusive mode. You can do this with the `nvidia-smi` utility.

Share Improve this answer Follow

answered Oct 17, 2011 at 17:02

Kamil Kisiel
12.4k 8 50 70

1 Thank you! That definitely helped me get it working so I gladly accept your answer... There's one downside to this solution though... it depends on what the users say they would use. If you can point me to some elegant solution where the gpu resource is actually monitored and the `-l` parameter is only used to schedule the job instead of also being used to "calculate" how many GPUs are left, that would be great :) – luxifer Oct 18, 2011 at 7:08

nevermind... I've been able to create a load sensor for this :) – luxifer Oct 18, 2011 at 9:02

Care to share your load sensor solution? Is there some output of nvidia-smi you can monitor to see how many GPUs are actually in use? or? – Kamil Kisiel Oct 18, 2011 at 20:21

I've already edited my original question to share that solution... the essence of it is: it only works if you put your GPUs in EXCLUSIVE_THREAD mode... then you'll get the number of *used* GPUs with the following command: `nvidia-smi |grep "Process name" -A 9|grep -v +-|grep -v \|=|grep -v Usage|grep -v "No running"|wc -l` – luxifer Oct 19, 2011 at 6:26

1 You can set the value of "requestable" to be FORCED instead of YES. Then only jobs that specify a value for gpu will be considered to run in the queue. Make sure you set the default to be NONE instead of 0. Also you should just not assign the gpu complex to the non-gpu queue at all. – Kamil Kisiel Jan 11, 2013 at 23:39

5

Open Grid Engine added GPU load sensor support in the 2011.11 release without the need for nvidia-smi. The output of the nvidia-smi application may (and does) change between driver releases, so the other approach is not recommended.

If you have the GE2011.11 source tree, look for: dist/gpu/gpu_sensor.c

To compile the load sensor (need to have the CUDA toolkit on the system):

% cc gpu_sensor.c -lnvidia-ml

And if you just want to see the status reported by the load sensor interactively, compile with:

-DSTANDALONE

To use the load sensor in a Grid Engine cluster, you will just need to follow the standard load sensor setup procedure:

http://gridscheduler.sourceforge.net/howto/loadsensor.html

Sources:

1. http://marc.info/?l=npaci-rocks-discussion&m=132872224919575&w=2

Share Improve this answer Follow

answered Feb 14, 2012 at 0:21

Robert Alexander
51 1 2

4

When you have multiple GPUs and you want your jobs to request a GPU but the Grid Engine scheduler should handle and select a *free* GPUs you can configure a RSMAP (resource map) complex (instead of a INT). This allows you to specify the amount as well as the names of the GPUs on a specific host in the host configuration. You can also set it up as a HOST consumable, so that independent of the slots your request, the amount of GPU devices requested with -l cuda=2 is for each host 2 (even if the parallel job got i.e. 8 slots on different hosts).

```
qconf -mc
    #name               shortcut   type        relop   requestable consumable default
urgency
    #------------------------------------------------------------------------------
--------------
    gpu                 gpu        RSMAP       <=      YES         HOST        0
0
```

In the execution host configuration you can initialize your resources with ids/names (here simply GPU1 and GPU2).

```
qconf -me yourhost
hostname              yourhost
load_scaling          NONE
complex_values        gpu=2(GPU1 GPU2)
```

Then when requesting -l gpu=1 the Univa Grid Engine scheduler will select GPU2 if GPU1 is already used by a different job. You can see the actual selection in the qstat -j output. The job gets the selected GPU by reading out the $SGE_HGR_gpu environment variable, which contains in this case the chose id/name "GPU2". This can be used for accessing the right GPU without having collisions.

If you have a multi-socket host you can even attach a GPU directly to some CPU cores near the GPU (near the PCIe bus) in order to speed up communication between GPU and CPUs. This is possible by attaching a topology mask in the execution host configuration.

```
qconf -me yourhost
hostname              yourhost
load_scaling          NONE
complex_values        gpu=2(GPU1:SCCCCScccc GPU2:ScccSCCCC)
```

Now when the UGE scheduler selects GPU2 it automatically binds the job to all 4 cores (C) of the second socket (S) so that the job is not allowed to run on the first socket. This does not even require the -binding qsub param.

More configuration examples you can find on www.gridengine.eu.

Note, that all these features are only available in Univa Grid Engine (8.1.0/8.1.3 and higher), and not in SGE 6.2u5 and other Grid Engine version (like OGE, Sun of Grid Engine etc.). You can try it out by downloading the 48-core limited free version from univa.com.

Share Improve this answer Follow edited Feb 5, 2013 at 16:00 answered Feb 5, 2013 at 7:10

Daniel
41 2

0

For SGE 2011.11 that comes with ROCKS 6.1 I found that setting the complex consumable to:

```
    #name               shortcut   type        relop   requestable consumable default
urgency
    #------------------------------------------------------------------------------
--------------
    gpu                 gpu        INT         <=      YES         JOB         0
0
```

This allowed me to set the number of GPUs per node and when I submitted a job the number of GPUs requested was not dependent on the SMP/SLOT count. I can then use 8 CPUs and 4 GPUs per job and not cause problems with other jobs leaking in. I still had to set the consumables for the nodes as above.

This is not as nice of a solution as some of the others but I found that the RSMAP option was not available in SGE 2011.11. I would like to eventually get this kind of configuration as I could then set which GPUs get used.

Hope this helps someone save a few hours of configuration.

Share Improve this answer Follow edited May 9, 2013 at 20:11 answered May 9, 2013 at 19:00

89c3b1b8-b1ae-11e6-b842-48d705
1,475 1 13 22

Thomas J Mustard
1