FRIDAY, JULY 24, 2015

VISIT MAP (THANK YOU)

REVOLVERMAPS

Installing and Setting Up Sun Grid Engine on a Single Multi-Core PC [tested on Ubuntu 16.04, 14.04, 12.04]

Note: this also works on:

Ubuntu 16.04 LTS,
tested on 05/25/2016
Linux s2 4.4.0-22-generic #40-Ubuntu SMP Thu May 12 22:03:46 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux

Ubuntu 14.04 LTS
test on 05/24/2016, 07/24/2014

INSTALLING AND SETTING UP SUN GRID ENGINE ON A SINGLE MULTI-CORE PC

Introduction

I was on the phone the other day with a friend of mine. He told me that he wanted to execute his C program for several sets of input arguments. His was planning to use his quad-core workstation at work. He wanted to ensure that the four cores would remain busy until all the runs of the program had completed for the various sets of input arguments and that each core would run for exactly one set of arguments at a time.

His initial thought had been to make use of the “make” command. This solution would entail a makefile, which would be invoked by running “*make -j4*” in the directory where the makefile would reside. The “-j4″ option specifies that four jobs should run simultaneously, thus allocating a single job at a time to each of the four cores.

I wasn't aware of such implementation of the "make" command. I found it elegant, although none of us dedicated the time to find out how the makefile should be written. My initial thought was to write a bash script so as to regulate the submission of jobs. However, the term "job scheduling" then came to my mind. It would be probably more time saving in the long run, more neat and it would offer better job supervision to employ a job scheduler.

The Load Sharing Facility (LSF) platform and Condor were the two job schedulers I knew from my work experience. My friend suggested the Son of Grid Engine (SGE). The obvious challenge was the installation and setting up of the job scheduler. There are numerous manuals, user guides, FAQs and forums explaining how to install SGE on a cluster. We have been inexperienced starters intending to use SGE to cover our individual computing needs on the basis of a single multic-core machine instead of a cluster. It turned out to be a cumbersome task for us to find coherent documentation for dummies to resolve the installation task at hand.

Before going any further with SGE, it is worth mentioning GNU Parallel, which is a shell tool for executing jobs in parallel in one or more computers. Parallel's scope is somewhat different to SGE's, since the former is not a fully fledged job scheduler. Nevertheless, Parallel is a lightweight tool, it is user-friendly, it handles job allocation to multiple cores of the same machine and it allows submission of multiple job arrays. In fact, my friend fulfilled his initial program execution request with Parallel. So, Parallel is an alternative functional tool for parallel job execution suiting the needs of an individual user.

In what follows, I will try to provide an overview on how to install and set up SGE on a single multi-core PC. As it usually happens, this post may not solve all the problems you might encounter during installation, although the intention has been to make it self-complete. It is the encounter of the exciting voyage of my friend and I, summoning our naive first-hand experience of tailoring the SGE installation to the modern reality of an individual user with a single multi-core machine.

The installation and setup instructions have been based on the Linux Mint Debian (LMDE) distribution and have been tested on Kubuntu 11.10. This guide is structured in three parts. The first part focuses on the installation, the second on setting up SGE and the third on testing SGE by means of elementary examples.

Part 1: Installation of SGE

To embark on the installation of the *gridengine* packages, run the following command on your terminal:

```
sudo apt-get install \
  gridengine-master gridengine-exec gridengine-common gridengine-qmon gridengine-client
```

Instead, you can run the shorter, and perhaps more error-prone, command

```
sudo apt-get install gridengine-*
```

A pop-up window will appear within the terminal during installation, with title “Configuring gridengine-common”. A series of questions show up sequentially in this window:

1. Question: “Configure SGE automatically?” Answer: highlight “<Yes>” and press “Enter”.
2. Question: “SGE cell name:” Answer: type “default”, then press “Tab” to highlight “<Ok>” and press “Enter”.
   Note here that you are free to choose any name you want for your SGE cell instead of “default”, such as “sge_cell” for example. If you alter the SGE cell name, you will have to subsequently set the *SGE_CELL* variable in your ~/.bashrc file accordingly (assuming that bash is your default shell). For instance, if you set the SGE cell name to be sge_cell, you will add the following line in your ~/.bashrc:

   ```
   export SGE_CELL="sge_cell"
   ```

   Furthermore, you will need to add the above line of code in your /root/.bashrc file so that the SGE cell is also known to the root. It is advised that you leave the SGE cell name as it is, holding the “default” value.

3. Question: “SGE master hostname:” Answer: type “localhost”, then press “Tab” to highlight “<Ok>” and press “Enter”.
   Instead of “localhost”, you can choose the hostname of your computer, which can by found by running the “hostname” command from the terminal:

   ```
   hostname
   ```

After answering these three questions, the pop-up window closes and the installation continues on the terminal. If for any reason you need to reconfigure the *gridengine-master* package, you can do so by invoking the following command:

```
sudo dpkg-reconfigure gridengine-master
```

The installation of gridengine is now complete, yet this does not mean that you are necessarily ready to use SGE. First of all, check whether *sge_qmaster* and *sge_execd* are running by using the command

```
ps aux | grep "sge"
```

The output I got verified that sge_qmaster and sge_execd are running:

```
sgeadmin  1310  0.0  0.1 135968  5376 ?        Sl   13:41   0:00 /usr/lib/gridengine/sge_qmaster
sgeadmin  1336  0.0  0.0  54760  1544 ?        Sl   13:41   0:00 /usr/lib/gridengine/sge_execd
1000      3171  0.0  0.0   7780   860 pts/0    S+   13:54   0:00 grep --colour=auto sge
```

If this is not the case for you, then start up sge_qmaster and sge_execd by executing the following three commands:

```
sudo su
sge_qmaster
sge_execd
```

Once you ensure that sge_qmaster and sge_execd are running, try to start *qmon*, the graphical user interface (GUI) for the administration of SGE:

```
sudo qmon
```

It is likely that the qmon window will not load, but instead you will get an error message. This is what I got:

```
Warning: Cannot convert string "-adobe-courier-medium-r-*--14-*-*-*-m-*-*-*" to type FontStruct
Warning: Cannot convert string "-adobe-courier-bold-r-*--14-*-*-*-m-*-*-*" to type FontStruct
Warning: Cannot convert string "-adobe-courier-medium-r-*--12-*-*-*-m-*-*-*" to type FontStruct
X Error of failed request:  BadName (named color or font does not exist)
  Major opcode of failed request:  45 (X_OpenFont)
  Serial number of failed request:  643
  Current serial number in output stream:  654
```

The error message indicates that some fonts are missing. The package which contains the necessary fonts is called *xfonts-75dpi*. In my case, xfonts-75dpi was installed automatically alongside the installation of the gridengine packages. Nevertheless, I got the error message because the fonts were not loaded after their installation. So, I merely restarted my computer. After rebooting, the “sudo qmon” command loaded the qmon window. If xfonts-75dpi is not installed on your system, then install it using the following command and then reboot:

```
sudo apt-get install xfonts-75dpi
```

After having resolved any possible font-related issues “sudo qmon” should load the SGE admin window. If you let the window remain idle or if you try to press any of its buttons, such as “Job Control”, the most likely event will be the appearance of a message pop-up window with the text “cannot reach qmaster”. Click on the “Abort” button of the pop-up window to terminate qmon. Try also the *qstat* command, which in my case gave the following error message:

```
error: commlib error: access denied (client IP resolved to host name "localhost". This is not identica
error: unable to contact qmaster using port 6444 on host "russell"
```

It is useful to delve in the error message in conjunction with the /etc/hosts file of my system:

```
127.0.0.1   localhost
127.0.1.1   russell
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

The hostname of my computer is “russell”. According to the error message, SGE set the client hostname to “russell”, whose LAN IP address is 127.0.1.1, while it set the client IP to 127.0.0.1, which is the LAN IP designated to the hostname “localhost”. To resolve this ambiguity, I changed the first two lines of my /etc/hosts so that both hostnames “localhost” and “russell” share the same LAN IP (as a word of warning, make a backup of your /etc/hosts file before making any changes to it). To be more specific, I deleted the second line and appended the “russell” hostname to the end of the first line. My /etc/hosts file thus became:

```
127.0.0.1   localhost russell
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
```

```
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Moreover, it is possible that your /etc/hosts file contains by default the string “localhost.localdomain” in the first line, for example as in

```
127.0.0.1   localhost localhost.localdomain russell
```

If that’s the case, make sure you remove “localhost.localdomain” so that only “localhost” and your machine’s hostname (“russell” is my hostname), are tied to the LAN IP 127.0.0.1:

```
127.0.0.1   localhost russell
```

You may restart sge_qmaster and sge_execd, although it is not advised given that you made a fundamental change to your system’s state by reconfiguring the association between IPs and hostnames in the /etc/hosts file. Instead, you are advised to restart your computer before you proceed any further. After rebooting, “qstat” and “sudo qmon” should run without returning any error messages.

Part 2: Setting up SGE

SGE is now (assumed to be) installed on your system and it needs to be set up before using it. To this end, it it will be demonstrated how to use qmon in order to initialize and thereafter administrate SGE.

1. Firstly, run “sudo qmon” and click on the “*User Configuration*” button of the loaded qmon window. On the text box of the “*User*” tab, insert your username, let’s say “*theodore*” (without the double quotes), and then click on the “Add” button. As soon as you add yourself as an SGE user, the “User” tab of the user configuration window will look similar to the following snapshot:

User tab of user configuration panel of qmon

Click on the “*Userset*” tab and either create a new set of users or highlight an existing one. To follow up with my exemplified setup, highlight the existing set of users called “arusers” and press “Modify”. A new window pops-up. In the text box with title “User/Group”, type the username of the SGE user that you recently created (theodore) and press “Ok”. The snapshot below shows how the “Userset” tab of the user configuration window will look:

Userset tab of user configuration panel of qmon

Press “Done” to leave the user configuration window.

2. Click on the “*Host Configuration*” button of the qmon window. On the text box of the “Submit Host” tab, insert the hostname, for example “localhost” (without the double quotes), and click on the “Add” button. This is a snapshot of the submit host tab of the host configuration panel:

Submit host tab of host configuration panel of qmon

Select the “Execution Host” tab of the host configuration panel. Highlight one of the hosts, say the localhost, and click on the “Modify” button. A new window pops up. Select the “*User Access*” tab, then highlight one of the access list entries (for example arusers) and, after making sure that the “Allow Access” radio button is chosen, click on the red right arrow as demonstrated in the snapshot below:

Giving access to a set of users for a selected execution host

Click “Ok” to return to the host configuration panel. This is how the localhost entry in the execution host tab of the host configuration panel will look:

Execution host tab of host configuration panel of qmon

> Notice that the “Access” entry of the “Access Attributes” list has the value “arusers”, not “NONE”.

Finally, click on the “*Queue Control*” button of the qmon window. Press “Add” under the “*Cluster Queues*” tab. A new window loads. Type the name of your choice for the new queue in the text box with title “Queue Name” (“mainqueue” was chosen for the current example). Type also the hostname (for example localhost) in the text box with title “New Host/Hostgroup” and then click on the red left arrow so that the hostname is added to the hostlist of the queue. On the “User Access” tab of the current window, highlight one of the access list entries (for example arusers) and, after making sure that the “Allow Access” radio button is chosen, click on the red right arrow. The following snapshot shows a snapshot of the “User Access” tab:

Giving access to a set of users for a selected queue

Select the “*General Configuration*” tab of the current window. Change the following entries:

- The value of the “Processors” entry is by default set to “UNDEFINED”. Set it to be equal to the number of processors of your PC. I set it to 2, as I have a dual-core computer.
- Specify the “Shell”. By default, it is set to /bin/csh. I set it to /bin/bash, as I prefer to work on the bash shell.

- Specify the number of “Slots”, which is by default set to 1. Should you wish, you can define more slots than the number of cores and regulate the maximum number of concurrently running jobs retrospectively using the “-tc” option for job arrays. I chose to set the number of slots to be as many as the cores of my computer, that is 2.

This is how the general configuration tab of the queue control panel will look after the above changes:

General configuration tab of queue control panel of qmon

Press “Ok” to leave the current window of the newly created queue, then click on “Done” to leave the queue control panel and finally click on the “Exit” button of the qmon window to exit qmon. SGE has been set up and is ready to be tested.

Part 3: Elementary Test Runs on SGE

As a matter of personal practice, I prefer to organize my submitted jobs with the help of wrapper scripts. Generally speaking, if the script containing the actual code is called scriptname.sh, I write another wrapper script called scriptname_qsub.sh which invokes qsub to submit scriptname.sh on SGE.

“Hello world” Test

To start with, submit a typical “hello world” bash script in order to check whether SGE works. This toy script simply echoes “Hello world” to a text file. Create a directory called hello_world. In the current example, its full path is

```
/home/theodore/tmp/hello_world
```

Select your own full path and modify subsequent paths accordingly. The hello_world folder contains two bash scripts, namely hello_world.sh and hello_world_qsub.sh. The former script holds the “echo” command

```
#!/bin/bash

echo "Hello world" > /home/theodore/tmp/hello_world/hello_world_output.txt
```

while the latter invokes the following "qsub" command

```
#!/bin/bash

qsub \
  -e /home/theodore/tmp/hello_world/hello_world_qsub.error \
  -o /home/theodore/tmp/hello_world/hello_world_qsub.log \
  ./hello_world.sh
```

The options "-e" and "-o" set respectively the paths used for the standard error and standard output stream of the job. While in the hello_world directory, submit the job by executing the hello_world_qsub.sh script:

```
./hello_world_qsub.sh
```

You will get some output similar to

```
Your job 1 ("hello_world.sh") has been submitted
```

You can use the "qstat" command to check the progress of the submitted job. As soon as execution completes successfully, three new files will be present in the hello_world folder:

- the file hello_world_output.txt will contain a single line with the string "Hello world",
- the file hello_world.error will be empty and
- the file hello_world.log will be empty.

Job Array Test

To confirm that submission of job arrays operates without complications, a relevant example is provided. The job array consists of five tasks, each of which echoes its ID, which is held by the environment variable *$SGE_TASK_ID*. Create a directory called "job_array", whose full path is

```
/home/theodore/tmp/job_array
```

Amend the path to fit your realization of the job array. Create two scripts in job_array, the job_array.sh and its wrapper counterpart job_array_qsub.sh. The contents of the former are

```
#!/bin/bash
echo "SGE_TASK_ID=$SGE_TASK_ID" > \
  /home/theodore/tmp/job_array/job_array_$SGE_TASK_ID.output.txt
sleep 3

```

while the contents of the latter are

```
#!/bin/bash
qsub \
  -e /home/theodore/tmp/job_array \
  -o /home/theodore/tmp/job_array \
  ./job_array.sh

```

From the jog_array directory, execute the wrapper script:

```
./job_array_qsub.sh
```

Note the double appearance of the $SGE_TASK_ID variable in job_array.sh, once as part of the name of the output files and once as a variable in the “echo” command. The resulting output will be five files job job_array_i.output.txt, for i=1, 2, 3, 4, 5, each containing a single line populated by the correpsonding string “SGE_TASK_ID=i”.

As for the job_array_qsub.sh script, notice that the argument to the “-e” and “-o” options is the full path of the job_array directory. Therefore, the standard error and output streams of the tasks of the job array will be saved in this directory. The default file name will be used, which follows the form job_name.ejob_id.task_id. In my run of the example, five empty standard error and five standard output (log) files were generated in hello_world_array, with names hello_world_array.sh.e23.[1-5] and hello_world_array.sh.o23.[1-5] respectively, where 23 happened to be the job ID at the time of execution.

A few closing remarks on the job array test run:

- The order of arguments to qsub can be critical. For instance, I made the mistake to enter the "-t 1-5″ option after the script invocation "./job_array.sh", which threw an error.
- If you want to throttle the number of concurrently running jobs, you can use the "-tc" options. For instance, if I had defined four slots on my dual-core machine with qmon and wanted to avoid having more than one running jobs in each core, I would have used the command

```
qsub -t 1-5 -tc 2 ./job_array.sh
```

- It is possible to overwrite the default file name for the standard error and standard output files of the job array, although the default naming convention is admittedly handy in several occasions. To facilitate the choice of custom file name, SGE offers the pseudo-environment variables *$JOB_ID*, *$TASK_ID*, *$JOB_NAME*, *$HOME*, *$USER* and *$HOSTNAME*. I tried to use the first four of them in the wrapper script job_array_qsub.sh, yet unfortunately they returned empty strings. I haven't found a solution for this issue so far. I rely on the default file name as a temporary solution.

"Hello world" Job for Testing email Notification

As a final example, the "Hello world" job is rerun with a couple more qsub options so as to send an email notification upon job completion. Mail delivery is achieved in this example by using the *exim4* message transfer agent. Follow the instructions of this post in order to set up exim4 to route emails through the Gmail SMTP servers. You can confirm that exim4 has been successfully configured with the help of a test "*mail*" command:

```
echo "Mail body: an exim4 test" | mail -s "Mail subject: exim4 test" "recipient@somewhere.com"
```

Once exim4 is set up, create a directory called "mail_test" and populate it with the mail_test.sh script

```
#!/bin/bash

echo "Hello world" > /home/theodore/tmp/mail_test/mail_test_output.txt
```

and with the mail_test_qsub.sh wrapper script

```
#!/bin/bash

qsub \
  -e /home/theodore/tmp/mail_test/mail_test_qsub.error \
  -o /home/theodore/tmp/mail_test/mail_test_qsub.log \
  -m e \
  -M "recipient@somewhere.com"
  ./mail_test.sh

```

The “e” argument of the “-m” qsub option instructs SGE to send the email at the end of the job.

From the mail_test directory, execute the wrapper script:

```
./mail_test_qsub.sh
```

If everything works well, you will receive an email notifying you that job execution is complete. I received an email with subject “Job 8 (mail_test.sh) Complete” containing the following lines:

```
Job 8 (mail_test.sh) Complete
 User             = theodore
 Queue            = mainqueue@localhost
 Host             = localhost
 Start Time       = 02/06/2012 21:02:43
 End Time         = 02/06/2012 21:02:43
 User Time        = 00:00:00
 System Time      = 00:00:00
 Wallclock Time   = 00:00:00
 CPU              = 00:00:00
 Max vmem         = NA
 Exit Status      = 0
```

As a word of warning, use the mail related qsub options with care. An erroneous submission of a single job or of a job array can potentially clutter your mailbox.

##########

ERROR FAQ:

How to solve the string conversion error:

Warning: Cannot convert string to type Pixmap

```
Warning: Cannot convert string "intro" to type Pixmap
Warning: Cannot convert string "toolbar_job" to type Pixmap
Warning: Cannot convert string "toolbar_queue" to type Pixmap
Warning: Cannot convert string "toolbar_submit" to type Pixmap
Warning: Cannot convert string "toolbar_cplx" to type Pixmap
Warning: Cannot convert string "toolbar_host" to type Pixmap
Warning: Cannot convert string "toolbar_cluster" to type Pixmap
Warning: Cannot convert string "toolbar_sched" to type Pixmap
Warning: Cannot convert string "toolbar_calendar" to type Pixmap
Warning: Cannot convert string "toolbar_user" to type Pixmap
Warning: Cannot convert string "toolbar_pe" to type Pixmap
Warning: Cannot convert string "toolbar_ckpt" to type Pixmap
Warning: Cannot convert string "toolbar_ticket" to type Pixmap
Warning: Cannot convert string "toolbar_prj" to type Pixmap
Warning: Cannot convert string "toolbar_rqs" to type Pixmap
Warning: Cannot convert string "toolbar_ar" to type Pixmap
Warning: Cannot convert string "toolbar_browser" to type Pixmap
Warning: Cannot convert string "toolbar_exit" to type Pixmap
```

You can ignore these warning since they are only related with font display. You can simply rely on the position on the GUI to make it right.

What if your submitted job is always in 'qw' status even there is enough slots:

Always use your pc's hostname to replace all the instance of "localhost" mentioned in the above documentation.

What is sge version in this testing?

Answer: 6.2u5-7.4

How to handle the connection error:

error: can't find connection

error: can't get configuration from qmaster -- backgrounding
Answer: ignore.

How to reconfigure:
To reconfigure the already installed after changing your hostname (for ubuntu in /etc/hostname) to match the one you got from running the above host command:
sudo dpkg-reconfigure gridengine-master

How to remove all SGE stuff to have a clean re-installation:
sudo apt-get purge gridengine-common gridengine-exec gridengine-client

Why there is only one job running in my SGE?
If you are the admin of SGE.
1) Type qmon
2) ==>Queue Control ==> Modify ===> General Configuration ===> Slots
put your CPU number there.

Note, the default value is 1.

```
% qstat -f
error: commlib error: got select error (Connection refused)
error: unable to send message to qmaster using port 6444 on host "<hostname>": got send error
%
```

When I upgrade ubuntu from 14.04 to 16.04, I got the above error.
My solution is to remove SGE and have a clean re-installation. (tested on 5/25/2016)

https://supcom.hgc.jp/english/utili_info/manual/faq.html

Reference:
[1] Note, the 90% contents of this post comes from:
Installing and Setting Up Sun Grid Engine on a Single Multi-Core PC | scidom:

[2]Howto set up SGE for CUDA devices? http://serverfault.com/questions/322073/howto-set-up-sge-for-cuda-devices

[3]
How to setup a single-machine (Sun) Grid Engine installation for unit tests on Travis-CI · GitHub:
'via Blog this'

Posted by AsrMan at 11:27 PM

Labels: CUDA, SGE, Sun Grid Engine, Ubuntu 14.04

No comments:

Post a Comment

To leave a comment, click the button below to sign in with Google.

SIGN IN WITH GOOGLE

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

FOLLOWERS

Followers (2)

Follow