Univa, An Altair Company

Grid Engine Documentation

Grid Engine Troubleshooting Quick Reference

Author:
Altair Engineering

Version:
8.7.0

September 6, 2021

Contents

1 Troubleshooting

1.1 User Troubleshooting

1.1.1 Problem: The output of Altair Grid Engine jobs, seems to be buffered, because looking into the jobs output file, it will be filled after jobs end.

Reason NFS implements loose caching, this means that file are written with a delay of a few seconds or after the writing has finished. It looks like Altair Grid Engine is buffering the output files but it is not. This is not a Altair Grid Engine behaviour, it's by NFS design.

Solution: Write your output files to a local filesystem instead of NFS.

1.2 Admin Troubleshooting

1.2.1 Problem: Problems with hostname resolving. Some hosts returning long, some short hostnames.

Reason Problems with hostname resolving is a very common problem which is often done at first Altair Grid Engine installations. In most cases it's a wrong setup and only in the rarest cases it was an Altair Grid Engine issue. In some cases host name resolution is configured to resolve using DNS and NIS (or other name resolution technique) and both name resolution approaches return different host names, for example DNS returns long name and NIS returns short name. When this happens Grid Engine may write corrupted configuration files because of the wrong host name resolution.

Solution: 1. Configure host name resolution in /etc/nsswitch.conf

```
- /etc/hosts
- nis
- dns
```

Configure the `hosts: db files nisplus nis dns` line in /etc/nsswitch.conf to resolve using the same mechanisms for all hosts in the cluster for example:

(configure dns and host files)

```
hosts: files dns
```

All hosts in the cluster must have the same name resolution mechanisms, and in the same order.

2. If you are using the files configuration:

```
- edit the /etc/hosts file.

do not map the hostname to the local host entry. like this: 127.0.0.1
```

```
localhost <your hostname>

192.168.1.10 qmaster_hostname.your-domain qmaster_hostname <- this is
optional
192.168.1.11 execd/submit/admin_hostname.your-domain execd/submit/
admin_hostname <- this is optional
192.168.1.12 execd1/submit1/admin1_hostname.your-domain execd1/
submit1/admin1_hostname <- this is optional
.
.
.
```

copy all entries into the /etc/hosts files of all your execd/submit/admin hosts

3. If you are using nis:

```
- execute the command:   ypcat -t hosts.byname and check the output if all
hosts of your cluster are in there and the hostnames are right and either
long or short.
```

4. If you are using dns:

```
execute the command: nslookup qmaster/submit/execd/admin_hostname and check
the output if all hosts of your cluster are in there and the hostnames are
right and either long or short.
```

5. Then check if all hosts are answering with the same hostname (long or short)

```
on the master host using gethostbyname -aname <qmaster hostname>
on the master host using gethostbyname -aname <execd/submit/admin hostname>
on execd/admin/submit host using gethostbyname -aname <qmaster hostname>
on execd/admin/submit host using gethostbyname -aname <execd/submit/admin
hostname>
```

Either ALL hosts in a cluster have to be resolved hosts as short hostnames (without domain) or ALL hosts in a cluster have to be resolved with long names (including the domain)

If you are using a mixed setup. eg a qmaster host with 2 network interfaces, were hosts from 2 different subnets submitting jobs to the cluster, then it's possible to setup a host_aliases file at: <sge_root>/<cell>/common/host_aliases

For documentation please look into the man page with: man host_aliases After that the hosts must be resolvable and return with the right hostnames. If this all is working then please remove the not working host and add it again. If possible eg. with test-systems and just a few hosts, a re-installation could be done.

1.2.2 Problem: What does “Skipping remaining x orders” mean, do I have to be concerned?"

You get error messages in your qmaster messages file looking like this:

01/25/2012 16:10:21|worker|grid-master|W|Skipping remaining 29 orders

01/25/2012 16:10:22|schedu|grid-master|E|unable to find job 4961392 from the scheduler order package

Reason:

The job with JOB_ID 4961392 could not be found due to job deletion or any error.

Solution: In case of this job has been deleted there is no reason for concerns otherwise it depends on the error messages at further checks why the job is gone have to be done.

If the job was deleted, this message is no error. So no solution can be provided.

1.2.3 Problem: Altair Grid Engine shows wrong m_socket counts for my execution hosts"

You are running a 2 socket machine with 4 cores each which should be reported with a topology:

SCCCCSCCCC but it’s reported like this: **SCSCSCSCSCSCSCSC**

Reason: Running an old kernel version. Kernel version < 2.6.16

Solution: Updating you kernel to a version 2.6.16 and higher

1.2.4 Problem: Problems with running large OpenMPI jobs in Altair Grid Engine >129 slots (or any other fixed slot count)"

You are running into the problem that when running jobs in Altair Grid Engine, it is not possible to request more then 129 slots. The request works, also Altair Grid Engine shows no error or crashes, just the job is hanging. Running this job outside of Altair Grid Engine also larger jobs > 129 slots are working.

Reason: The mpirun command looks like that:

```
mpirun -mca orte\_rsh\_agent ssh:rsh -mca ras\_gridengine\_verbose 100
-server-wait-time 60  /path/to/job/binary
```

The mcs_orte_rsh_agent parameter is set to ssh:rsh which might be the problem module here. OpenMPI seems to limit the number of concurrent rsh processes in this module.

Without Altair Grid Engine, the module is not loaded and no limit is set. More then 129 task are running. Used with Altair Grid Engine OpenMPI loads some additional modules setting this limit set and jobs using more the 129 slots won’t run.

Solution: To check this limit use the ompi_info command:

```
% ompi_info -all | grep plm_rsh_num_concurrent
```

MCA plm: parameter “plm_rsh_num_concurrent” (current value: “128”, data source: default value) <—- here 128 is set.

To workaround this problem the plm_rsh_num_concurrent parameter ca be set to be used by the mpirun call:

```
mpirun -mca plm_rsh_num_concurrent 256 -np 2000   <---- here set to 256
```

Setting this parameter is fixing the problem.

1.2.5 List of Altair Grid Engine Error and Exit Codes

The following table lists the job-related error codes or exit codes. These codes are valid for every type of job.

Table 1: Job related Error and Exit Codes

Script/Method	Exit or Error Code	Consequence
Job Script	0	Success
	99	Requeue
	All other values	Success: exit code in accounting file
prolog/epilog	0	Success
	99	Requeue
	All other values	Queue error state, job re queued

The following table lists the consequences of error codes or exit codes of jobs related to parallel environment (PE) configuration.

Table 2: Parallel-Environment-Related Error or Exit Codes

Script/Method	Exit or Error Code	Consequence
pe_start	0	Success
	All other values	Queue set to error state, job re queued
pe_stop	0	Success
	All other values	Queue set to error state, job not re queued

The following table lists the consequences of error codes or exit codes of jobs related to queue configuration. These codes are valid only if corresponding methods were overwritten.

Table 3: Queue-Related Error or Exit Codes

Script/Method	Exit or Error Code	Consequence
Job starter	0	Success

Script/Method	Exit or Error Code	Consequence
	All other values	Success, no other special meaning
Suspend	0	Success
	All other values	Success, no other special meaning
Resume	0	Success
	All other values	Success, no other special meaning
Terminate	0	Success
	All other values	Success, no other special meaning

The following table lists the consequences of error or exit codes of jobs related to checkpointing.

Table 4: Checkpointing-Related Error or Exit Codes

Script/Method	Exit or Error Code	Consequence
Checkpoint	0	Success
	All other values	Success. For kernel checkpoint, however, this means that the checkpoint was not successful.
Migrate	0	Success
	All other values	Success. For kernel checkpoint, however, this means that the checkpoint was not successful. Migration will occur.
Restart	0	Success
	All other values	Success, no other special meaning
Clean	0	Success
	All other values	Success, no other special meaning

For jobs that run successfully, the `qacct -j` command output shows a value of 0 in the failed field, and the output shows the exit status of the job in the exit_status field. However, the shepherd might not be able to run a job successfully. For example, the epilog script might fail, or the shepherd might not be able to start the job. In such cases, the failed field displays one of the code values listed in the following table.

Table 5: Job-Related Error or Exit Codes

Code	Description	acctValid	Meaning for Job
0	No failure	t	Job ran, exited normally
1	Presumably before job	f	Job could not be started
3	Before writing config	f	Job could not be started
4	Before writing PID	f	Job could not be started
5	On reading config file	f	Job could not be started
6	Setting processor set	f	Job could not be started
7	Before prolog	f	Job could not be started
8	In prolog	f	Job could not be started
9	Before pestart	f	Job could not be started

Code	Description	acctValid	Meaning for Job
10	In pestart	f	Job could not be started
11	Before job	f	Job could not be started
12	Before pestop	t	Job ran, failed before calling PE stop procedure
13	In pestop	t	Job ran, PE stop procedure failed
14	Before epilog	t	Job ran, failed before calling epilog script
15	In epilog	t	Job ran, failed in epilog script
16	Releasing processor set	t	Job ran, processor set could not be released
24	Migrating (checkpointing jobs)	t	Job ran, job will be migrated
25	Rescheduling	t	Job ran, job will be rescheduled
26	Opening output file	f	Job could not be started, stderr/stdout file could not be opened
27	Searching requested shell	f	Job could not be started, shell not found
28	Changing to working directory	f	Job could not be started, error changing to start directory
100	Assumedly after job	t	Job ran, job killed by a signal

The Code column lists the value of the failed field. The Description column lists the text that appears in the `qacct -j` output. If `acctValid` is set to `t`, the job accounting values are valid. If `acctValid` is set to `f`, the resource usage values of the accounting record are not valid. The Meaning for Job column indicates whether the job ran or not.

1.2.6 Troubleshooting problems with Docker jobs

There is a wide variety of problems that can happen with Docker and Docker jobs. Here is a description how to analyze and fix the most common ones.

Docker is not properly installed

To check if Docker is properly installed, login as the **Altair Grid Engine admin user** to that execution host and run

```
$ docker version
```

If the output is *"command not found"*, Docker seems to be not installed properly. If the package manager of that host says Docker is installed, check if the `$PATH` environment variable points to the `docker` binary and/or reinstall the Docker package.

If only the client version, but not the server version is printed, Docker is installed but the Docker daemon cannot be contacted, which could be caused by:

- A firewall preventing the communication
- The Altair Grid Engine admin user is not member of the "docker" group on that execution host

- The Docker daemon is not running - check using
  `$ ps aux | grep dockerd` - see next paragraph for a solution

If both the client and the server version are printed, then Docker is installed, the Docker daemon is running and can be contacted.

Docker is installed but the daemon is not running

Some package managers start the Docker daemon after installation, some do not. Some add and enable startups scripts for the Docker daemon, some do not. With `systemd` the startup scripts are enabled and the daemon is started using these commands:

```
$ sudo systemctl enable docker
$ sudo systemctl start docker
```

Check if these `systemctl` calls report any errors! Then, as the Altair Grid Engine admin user, run

```
$ docker version
```

again to check if the Docker daemon runs fine now.

The Docker daemon does not work properly

Check if the Docker daemon works properly:

- Check if Docker can start two different kinds of containers:
  - `$ docker run hello-world:latest`
    If it is not available locally, it first prints how it downloads the image.
    The container itself just prints some text and quits then.
  - `$ docker run -it ubuntu:latest`
    `root@f4e8d6368c0f:/#`

If it fails to run these containers, reinstall Docker.

- Check if the list of available images looks OK:
  `$ docker images`
  The "<null>" entries are anonymous intermediate image layers and no error.
  If there are broken images, try to delete them with the command:
  `$ docker rmi <image>`
- Check if running or finished Docker containers could cause problems:
  `$ docker ps` (shows only running containers)
  `$ docker ps -a` (shows running and finished containers)

Unclear job error reason and `docker run` output

The job error reason can contain a message like this and `docker run` print the same error message:

```
invalid header field value "oci runtime error: container_linux.go:247:
starting container process caused \"process_linux.go:359:
container init caused \\\"rootfs_linux.go:53:
mounting \\\\\\\"cgroup\\\\\\\" to rootfs \\\\\\\"/var/lib/docker/devicemapper
```

```
/mnt/8c26d2f3ae0b4d69b4375705d3c5b03386e64c9cec69dd012e972f2055acf820/rootfs\\\\\\\"
at \\\\\\\"/sys/fs/cgroup\\\\\\\" caused \\\\\\\"no subsystem for mount\\\\\\\"\\\"\"
```

This is caused by a broken Docker installation, solutions to this problem can be found on the internet, but normally reinstalling Docker is the quickest solution.

Altair Grid Engine does not detect Docker

If `qstat -F docker` does never report `docker=1` for an execution host that should be a Docker host and Docker itself works fine on that execution host, first check if the Altair Grid Engine version you are using supports the Docker version of this execution host. The list of supported Docker versions can be found in the ReleaseNotes.

Then, like any other loadvalue, the `docker` and `docker_images` values could be overwritten by a fixed value in the host or queue configuration. Use `qconf -se <host>` and `qconf -sq <queue>` to check the configured `complex_values`.

Altair Grid Engine does not detect Docker images

If `qstat -F docker_images` does never report Docker images, while there are some available on the that execution host and `docker=1` is reported for that execution host, first run
`$ docker images`
on that execution host as the Altair Grid Engine admin user to see if this reports the images. If not, possibly there is some information in the system log or the Docker daemons log file, if one is configured. It depends on the Linux distribution and on start parameters to the Docker daemon where it logs to.

Docker jobs fail to start

If the job or execution host is in "E" state and the job script itself was not started,
`$ qstat -j <jobid>`
should print an error reason.

These are possibly not obvious:

- "409 - Conflict":
  Altair Grid Engine gives the containers it starts specific names in the format "UGE_job_5.1". If a container with such a name already exists (running or finished), the Docker daemon does not create a second one, so the job fails.
- "500 - Internal Server Error":
  This is a general error, typical corresponding error message is "Device is Busy" which means the Docker daemon cannot create two containers from the same image at the same time. Altair Grid Engine does retry to create a container from that image, but not ad infinitum, so if the Docker image is always busy, the job may fail eventually with this error message.

Debugging the communication between Altair Grid Engine and the Docker daemon

From Altair Grid Engine 8.6.5 on both the execution daemon and the job shepherd log to their `messages` resp. `trace` file what they send to the Docker daemon and what they receive from it.

The execution daemon logs the communication to its `messages` file when the `loglevel` is set to `log_debug`. This can produce a lot of output! The job shepherd always logs the communication to its `trace` file which is normally deleted after job end, but kept if the `execd_params`

setting KEEP_ALIVE is set to always or true. Here, the communication usually produces not so much output.

To see the whole communication from the Docker daemons point of view, first stop the Docker daemon:

```
$ sudo systemctl stop docker
```

and then start it in debug mode as user “root”:

```
$ /usr/bin/dockerd -D
```

It prints information like this:

```
DEBU[2018-02-27T18:54:57.010682419+01:00] Calling POST /containers/create
DEBU[2018-02-27T18:54:57.010954063+01:00] form data: {"AttachStderr":true,
"AttachStdin":false,"AttachStdout":true,"Cmd":["-cm"],
"Entrypoint":"/uge_mnt/work/master/cluster/bin/lx-amd64/sge_container_shepherd",
"Env":null,"HostConfig":{"Binds":["/var:/uge_mnt/var","/work:/uge_mnt/work",
"/home:/uge_mnt/home","/work:/work","/work:/work"],
"RestartPolicy":{"MaximumRetryCount":0,"Name":""}},
"Image":"ubuntu:14.04","Labels":{"com.univa.gridengine.cell":"default",
"com.univa.gridengine.job_number":"15","com.univa.gridengine.root":"/work/master/cluster"},
"OpenStdin":false,"StdinOnce":false,"Tty":false,"User":"1000:100",
"WorkingDir":"/uge_mnt/var/spool/testsuite/4104/execd/polaris/active_jobs/15.1"}
```

In this example, the bind “/work:/work” was submitted two times which is not allowed, so the job was set to error state.