Interactive and Batch Jobs

HP XC4000 Cluster

Horst Gernert
University of Karlsruhe
Computing Center

Gernert@rz.uni-karlsruhe.de
Software Environment for Running (MPI) Programs

» We build our own environment: Job Management System (JMS). The XC batch user interface as described in the original HP Documentation is not accessible for the users on our system.

» Based on HP MPI ( mpirun ) and Slurm.

» SLURM ( Simply Linux Utility for Resource Management) is part of the CHAOS project, developed at the Lawrence Livermore National Laboratory (LLNL).
Model of Operation

- Run program interactive (Login Nodes).
- Run program in batch mode (Batch Nodes).
  - Development and test (development class)
    - Partition d (4 nodes)
    - Resources are shared (nodes, CPUs, memory)
  - Production (production class)
    - Partition t (750 nodes)
    - Nodes used exclusively
Software Environment for running (MPI) programs

User Commands:

» job_run (alias mpirun alias mpirun.mpich)
  – Run an mpi program.

» job_submit
  – Create a batch job.

» job_info
  – Show limits and other JMS informations.

» job_queue
  – Show all batch jobs of the user.

» job_cancel
  – Delete a job.
General Behaviour of \texttt{job...} Commands

\begin{itemize}
\item Option \texttt{–h} prints a short syntax help.
\item Option \texttt{–H} prints a detailed description of the command (like \texttt{man}).
\item Messages go to STDOUT.
\item Errors go to STDERR.
\item If the command failed, the return code is \textgreater{} 0.
\end{itemize}
job_run / mpirun

job_run { | -? | -h[elp] | -H }

[-lsided] [-mpich]
[-i <spec>] [-np <tasks> | -n <tasks>]
[-distribution={block|cyclic}]
[-cpu_bind[_mt]=[v,][option][,v]]
[-stdio=<options>]

{ -f <appfile> | program }

mpirun is equivalent to job_run are equivalent.

mpirun.mpich is equivalent to job_run -mpich ...
job_submit

job_submit {  | -? | -h | -H }

job_submit -t time -m mem -c class -p i[/j] job

Parameter:

- -h: help
- -?: help
- -H: more help
job_submit  (options)

-t time: maximum CPU time per thread / CPU (minutes)
-T time: maximum elapse time of the job (minutes)
-m mem: maximum real memory requirement per process / task
-p i[/j]: number of processes (tasks) i [ and threads per task j ]
-c: job class: class
   class=d: development class
   class=p: production class

job: name of the executable (program / shell script / ...)
     if you want to add parameters to the job
     you can write them after the job name.
job_submit (examples)

Environment:

Parameters can also be set by "export JMS_<parameter>=value"
Examples: JMS_t=time; JMS_o=file; JMS_job=job
Parameters set in the command line overwrite parameters set by the environment. job_submit itself set the environment and exports it to the user job.

Examples:

job_submit -p 4 -c d -t 20 -m 100 test_job -x 1000 5 16 0
job_submit -p 1 -cp -t 2000 -T 3000 -m 2000 big_serial_job
job_submit -p 64 -c p -t 2000 -m 1000 big_parallel_job 4 10
job_submit -p 8/2 -cp -t 200 -m 6000 2_threads_per_task
job_submit (additional options)

- `i file:` standard input (default: /dev/null)
- `o file:` standard output (default: Job_\$JID.out)
- `e file:` standard error (default: Job_\$JID.err)
  If file = + then stderr = stdout ( 2>&1 )

- `J name:` job name (write it in quote marks: "job name")
job_submit  (additional options)

-N notify  Notification (send mail by event)

notify := N[:Nadr]

N (notification mode):
  s  submit
  b  begin
  c  complete
  C  complete (with STDOUT and STDERR)
  e  error
  E  error (with STDERR)

Use only one of the options c, C, e  or  E !!!

Nadr (notification  /  mail adress) (default: $USER)
job_submit (additional options)

- l opt: list options:
  opt=a|A : list system account information (on|off)
  (default=on)

System data for job 26000

Number of CPUs .................................................................: 216
Sum of CPU-time over all processors (in secs) ......................: 3:00:09.202
Elapsed time (in secs) ..........................................................: 52.18
Maximum physical memory by any process (in MB) ...............: 1728
Maximum virtual memory by any process (in MB) ..................: 2253
Maximum number of minor page faults for any process ........: 5249527
Total number of voluntary context switches for all processes : 1748349
Status is ...............................................................: COMPLETED
Error variable has been set to ......................................: 0
job_info / job_queue

job_info

job_info gives you information about the limits of the job classes.

job_queue [ -l ]

- l  long output

List all jobs of the user in JMS job queue.
### job_info (part of output)

<table>
<thead>
<tr>
<th>c</th>
<th>q</th>
<th>i</th>
<th>j</th>
<th>m (Mbyte)</th>
<th>t (minutes)</th>
<th>T (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>max.</td>
<td>max.</td>
<td>max.</td>
<td>max.</td>
<td>max.</td>
<td>default max.</td>
<td>default max.</td>
</tr>
</tbody>
</table>

| d | 1 | 16 | 8 | 2000 | 60 | 2*t | 240 |
| p | 10 | 1024 | 4 | 16000 | 4320 | 1.01*t | 5400 |

- $i \times j \leq 1024$ \& $m \leq 4000$ $\rightarrow$ $i \leq 1024$
- $4000 < m \leq 8000$ $\rightarrow$ $i \leq 512$
- $8000 < m \leq 16000$ $\rightarrow$ $i \leq 256$
### job_queue (output)

<table>
<thead>
<tr>
<th>job-id</th>
<th>c</th>
<th>P</th>
<th>n/i/j</th>
<th>t</th>
<th>T</th>
<th>m</th>
<th>queued</th>
<th>s</th>
<th>start</th>
<th>end(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13652</td>
<td>d</td>
<td>d</td>
<td>1/1/1</td>
<td>30</td>
<td>60</td>
<td>200</td>
<td>04/13:21</td>
<td>r</td>
<td>4/13:21</td>
<td>4/43:21</td>
</tr>
<tr>
<td>13649</td>
<td>p</td>
<td>t</td>
<td>23/92/1</td>
<td>1200</td>
<td>1213</td>
<td>4000</td>
<td>04/03:34</td>
<td>r</td>
<td>4/09:07</td>
<td>5/05:07</td>
</tr>
<tr>
<td>13650</td>
<td>p</td>
<td>t</td>
<td>13/25/1</td>
<td>1800</td>
<td>1819</td>
<td>8000</td>
<td>04/08:14</td>
<td>r</td>
<td>4/12:01</td>
<td>5/18:01</td>
</tr>
<tr>
<td>13651</td>
<td>p</td>
<td>t</td>
<td>4/8/2</td>
<td>2400</td>
<td>2600</td>
<td>3000</td>
<td>04/08:27</td>
<td>r</td>
<td>4/12:01</td>
<td>6/00:01</td>
</tr>
<tr>
<td>13654</td>
<td>p</td>
<td>t</td>
<td>8/8/1</td>
<td>4300</td>
<td>4730</td>
<td>9000</td>
<td>04/09:09</td>
<td>w</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13563</td>
<td>p</td>
<td>t</td>
<td>16/8/2</td>
<td>240</td>
<td>250</td>
<td>6000</td>
<td>03/23:01</td>
<td>Lw</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>
job_cancel

job_cancel -jid [jid ...]

jid: Job identifier (first column of job_queue listing). You can only address your own jobs ;-).

job_cancel deletes one or more jobs from the job queue.

If the job is running, the job is killed.
Batch Jobs

```
job_submit ... myjob.bash

#!/bin/bash
....

job_run    myprogram
....
```

Environment and current directory are taken over to the user job.

Additional environment variables are set by JMS.
If the jobscribt contains only one command line, it can be ommited:

```bash
job_submit  -p1  ....  myprogram
job_submit  -p?  ....  job_run myprogram  ? > 1
```

$TMP is set by JMS and points to a local filesystem for each node, NOT for each task !!!
Batch Jobs (chaining)

```
#!/bin/bash

job_submit .... myjob.bash

if ....
  then
    job_submit
  fi

exit
```
Thank You!