PRACE – BOLT project

Elena Breitmoser, EPCC

April-June 2013

The original code was taken from the github-repository <https://github.com/aturner-epcc/bolt>. Any updates can be found in the ‘develop’-branch of <https://github.com/ebreitmo/bolt>.

To develop the bolt-script, we had access to the following PRACE architectures:

* FERMI, CINECA [1]
* HECToR, EPCC [2]
* Stokes, ICHEC [3]
* SupermUC, LRZ [4]

The bolt tool was extended to be able to generate MPI, OpenMP and hybrid batch scripts for each of the above architectures (the version for HECToR already existed), then the three versions of the test code ‘Hello World’ (see section 2) were run on each of them. The entries in green in Table 1 show successful results. Pure OpenMP jobs were not run on FERMI because the minimum number of nodes is 128. It was considered a waste of resources to test this case. Though, the tool is not restrictive; if users want to use less than they are charged for it will be up to them.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Stokes** | **SuperMUC** | **FERMI** |
| **Hyperthreading** | 12/24 cores | 16/32 cores | 16/64 cores |
| **Serial jobs** | Individual serial jobs not allowed  (taskfarm option) | Presently not available | Only on front-end node |
| **MPI jobs** | <=516 cores, <=84 hrs  mpiexec, qsub | <=512 nodes, <=48 hrs  mpiexec, llsubmit | 128-2048 nodes, <= 24 hrs  runjob, llsubmit, --exe a.out |
| **OpenMP jobs** | ./a.out | ./a.out | Probably not advisable (waste of resources) |
| **Hybrid jobs** | mpiexec  -npernode | mpiexec, export MP\_TASK\_AFFINITY=core:$OMP\_NUM\_THREADS  export MPSINGLE\_THREAD=no | runjob,  --exp-env OMP\_NUM\_THREADS --exe a.out |

Table : Three PRACE architectures and their special features regarding serial, MPI, OpenMP-and hybrid jobs.

For the reasons listed in Table 1, serial batch jobs on Stokes, SuperMUC and FERMI were set to ‘no’.

# Updating of the documentation

Both the user guide and the admin guide have been extended and updated.

The admin guide now describes all configuration files, i.e. the ‘resource’-, the ‘batch’-, the ‘global’- and the ‘codes’-file. It lists and describes all sections and options required for each of them.

The user guide includes examples for serial, MPI, OpenMP and hybrid job scripts as well as specific simulation codes (CP2K). Job scripts generated on a variety of architectures and for a variety of options, which the user can specify when generating a batch script , are shown.

# Simple test code

The ‘Evaluation on ISTP’ on the PRACE Wiki was extended.

Simple ‘Hello-World’-test codes are now available from there [6] for MPI, OpenMP and hybrid code. Examples are provided for generating a suitable batch script on HECToR. The resulting batch script is also shown.

# Extensions to the original bolt code

The added extensions described in the following, allow for more machine-tailored specifications and flexibility in generating a batch script. The added exception tests should allow for cleaner and easier-to-understand and repair error messages.

## Try/Catch clauses

We introduced try/catch clauses in ‘/modules/resource.py’ and ‘/modules/batch.py’ to ensure that all data read in from the ‘/configuration/resources/\*.resource’-and ‘/configuration/batch/\*.batch’-files regarding the sections and options and a value (if required) are both correct and complete. So, if a user changes an existing file incorrectly (regarding the file syntax and necessary entries, not the machine specifics) an error message will tell the user where what error occurs. The same applies for a user adding a new resource-or batch-file from scratch.

## Unittests

For the unittests, ‘testDistribution.py’ was updated to display the newly built-in features of the code correctly.

## Hyperthreading

Several architectures support hyperthreading. In this case the number of logical cores per node seen by the OS equals the number of physical cores multiplied by hardware threads per core. For example, on Stokes each physical core can appear as two logical cores. To account for this, ‘bolt’ will now choose the physical number of cores by default. But if a number of threads with

‘number of physical cores’ < number of threads asked for <= number of logical cores’,

is chosen by the user, this number will be used (also see section 1.1).

## Distinction between various kinds of parallel jobs

The original section ‘parallel jobs’ in ‘configuration/resources/\*/resource’ was replaced by ‘general parallel jobs’, and the three further sections ‘distributed-mem jobs’, ‘shared-mem jobs’ and ‘hybrid jobs’ were added. This allows adding a job launcher, additional job options, as well as script preamble and postamble commands specific to the kind of parallel job required, as they often differ for MPI and OpenMP code. The option ‘executable job options’ was added to account for example for the ‘—exe’ argument of the job launcher ‘runjob’ on FERMI.

## Serial job launcher

To cover the possibility that a serial batch job requires a job launcher, the option ‘serial job launcher’ was added (see section 1.3). For example, this is required for a simple (non-hybrid MPI/GPU) GPU job on Curie (see [5] and /configuration/resources/Curie-GPU.resource’).

## Summary of added resource options and sections

The following sections and options were added to the /configuration/resources/\*.resource-files. The classes /modules/resources.py, /modules/job.py and bin/bolt.py were modified accordingly to read in the new information and deal with it as required. A complete list and description of all sections and options can be found in the admin guide.

### Hyperthreading

The option ‘threads per core’.

## Distinguish between MPI/OpenMP/Hybrid characteristics

The sections ‘distributed-mem jobs’, ‘shared-mem jobs’, ‘hybrid jobs’, each with the options ‘parallel job launcher’, ‘additional job options’, ‘script preamble’, ‘executable job options’ and ‘script postamble’.

## Serial job launcher

The option ‘serial job launcher’.

# Future work

Unittests could be extended for the job-class and to check for overall functionality.. E.g. testing

* + 1. if more OpenMP threads than in node are requested.
    2. If serial job behaves as expected.

For the resource- and batch-classes exception clauses were used instead of unittests.

# References

1. IBM FERMI User Guide, <http://www.hpc.cineca.it/content/ibm-fermi-user-guide> (2013)
2. Running jobs on HECToR, <http://www.hector.ac.uk/support/documentation/userguide/batch.php> (2013)
3. ICHEC, Srokes and Stoney Documentation, <http://www.ichec.ie/support/documentation/> (2013)
4. SuperMUC Petascale system, <http://www.lrz.de/services/compute/supermuc/> (2013)
5. Best Practice Guide – Curie – [PRACE Research Infrastructure], <http://www.prace-ri.eu/Best-Pratcice-Guide-Curie-HTML?lang=en>

1. [Evaluation Of The Bolt Tool ForI STP](https://prace-wiki.fz-juelich.de/bin/view/PRACE/Operations/EvaluationOfTheBoltToolForISTP" \o "Topic revision: 7 (2013-04-22 - 13:39:32)), <https://prace-wiki.fz-juelich.de/bin/view/PRACE/Operations/EvaluationOfTheBoltToolForISTP>