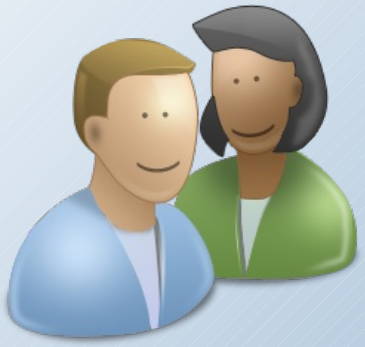
Middleware support to MPI through gLite, ARC and UNICORE

Dr Ivan Degtyarenko
NDGF / CSC – IT Center for Science, Finland

EGI Technical Forum 2010

NDGF
NORDIC DATAGRID FACILITY

MPI job through ARC



User: (i) binaries, (ii) the .xrsl script with a CPU number and wanted runtime environment, (iii) shell script to be executed on CE

ARC Client: discover the resources, brokering

ARC CE: run the runtime environment script, execute the job

Runtime Environment in general

http://www.nordugrid.org/applications/environments/

Runtime Environment Registry at CSC: http://gridrer.csc.fi/

- can be specified for any pre-installed application or environment
- typical usage by large research groups having deal with particular set of software
- by sysadmin: setup script (a Bash script) on the Computing Resource named after the environment (e.g. `MYSOFT-v2.0`), and placed in a dedicated directory
- by user: the end user defines the RE in the job description file as `(runTimeEnvironment=MYSOFT-v2.0)`

RTE for MPI in practice

RTE directory defined in **`arc.conf`** and can be any

```
[grid-manager]
runtimedir="/grid/arc/runtime"
```

Path to a particular MPI RTE: flavor/compiler+bitness

```
/grid/arc/runtime/ENV/MPI/OPENMPI-1.3/GCC64
```

RTE script is called by ARC with argument 0, 1 or 2.

- 0: made before the the batch job submission script is written
- 1: just prior to execution of the user specified executable
- 2: "clean-up" call, after the user's executable has returned

See the Bash script example for 64-bit OpenMPI on cluster with SGE

RTE script for MPI at ARC CE

.../ENV/MPI/OPENMPI-1.3/GCC64

```
#!/bin/bash
parallel_env_name="openmpi"
case "$1" in
0 ) # local LRMS specific settings
    i=0
    eval jonp=\${joboption_nodeproperty_$i}
    while [ ! -z $jonp ] ; do
        (( i++ ))
        eval jonp=\${joboption_nodeproperty_$i}
    done
    eval joboption_nodeproperty_$i=$parallel_env_name
;;
1 ) # user environment setup
    export MPIHOME=/home/opt/openmpi-1.3
    export PATH=$MPIHOME/bin/:$PATH
    export LD_LIBRARY_PATH=$MPIHOME/lib:$LD_LIBRARY_PATH
    export MPIRUN='mpirun'
    export MPIARGS="-v -np $NSLOTS"
;;
2 ) # nothing here
;;
* ) # everything else is an error
    return 1
;;
esac
```

User's files

openmpi.xrsl

```
&(jobName="openmpi-gcc64")
(count="4")
(wallTime="10 minutes")
(memory="1024")
(executable="runopenmpi.sh")
(executables="hello-ompi-gcc64.exe" "runopenmpi.sh")
(inputfiles=("hello-ompi-gcc64.exe" ""))
(stdout="std.out")
(stderr="std.err")
(gmlog="gmlog")
(runtimeenvironment="ENV/MPI/OPENMPI-1.3/GCC64")
```

runopenmpi.sh

```
#!/bin/sh
echo "MPIRUN is '$MPIRUN'"
echo "NSLOTS is '$NSLOTS'"
$MPIRUN -np $NSLOTS ./hello-ompi-gcc64.exe
```

MPI job running: show time

degtyare@bombay:~/egitf - Shell - Konsole <2>

Session Edit View Bookmarks Settings Help

```
// Hello world in C with MPI

#include <stdio.h>
#include <mpi.h>
int main(int argc, char **argv){
  int npes, irank, ierr;
  MPI_Init(&argc, &argv);
  ierr = MPI_Comm_size(MPI_COMM_WORLD, &npes);
  ierr = MPI_Comm_rank(MPI_COMM_WORLD, &irank);
  printf("r%2d: My rank is %d, number of total processes is %d\n",irank,
irank,npes);
  MPI_Finalize();
  sleep(1);
  printf("r%2d: Done.\n",irank);
  return 0;
}
~
~
~
~
~
~
~
~
~
~
:set number                                          1,1           All
```

Shell

ARC roadmap for MPI support

development is fully aligned with the EMI project, objectives include:

- better multi-core support on all emerging architectures resources
- multi-node execution on interconnected clusters
- scenarios for advanced topologies, FPGAs, GPGPUs
- common MPI execution framework, a "backend" across the different computing services to allow users to execute parallel applications in a uniform way

Finnish M-grid statistics

Number of jobs		Walltime (hours)	
total	6213569	total	48920078
serial	4753250 (76.50%)	serial	9535418 (19.49%)
parallel	1460319 (23.50%)	parallel	39384660 (80.51%)
lam	56640 (3.88%)	lam	2616030 (6.64%)
mpich	888456 (60.84%)	mpich	11557678 (29.35%)
mpich2	51152 (3.50%)	mpich2	889372 (2.26%)
openmpi	349598 (23.94%)	openmpi	11481871 (29.15%)
mvapich	79519 (5.45%)	mvapich	12234506 (31.06%)
threaded	31385 (2.15%)	threaded	100057 (0.25%)

Majority of the jobs are serial in job numbers but parallel (!) in terms of CPU time consuming

In terms of MPI

ability to run and compile MPI easily

the default recommended flavor (OpenMPI ?)

ability to request the varying number of slots

ability to request logical CPUs within

one physical CPU only, or one WN

available memory per logical CPU

interconnecting choice