ARC Information system LRMS modules overview

NorduGrid Conference 2015

2-5 June, Bern, Switzerland

Florido Paganelli florido.paganelli@hep.lu.se

NORDUGRID
Grid Solution for Wide Area Computing and Data Handling

Outline

- LRMS modules structure
- Specific LRMS modules
- Supported(?) LRMS plugins
- Commands run by each LRMS plugin

LRMS modules structure

- A PERL module-based plugin system.
- The main interface is defined in LRMSinfo.pm
  - Interface here is:
    - Two perl subroutines that must be defined in the plugin to return the correct information:
    - Two data structures that will be filled by the above subroutines. The format of the data structure is defined in this module.
- The data structure is checked in: InfoChecker.pm
  - I don't think this module is complete.

Specific LRMS modules

- Of two kinds:
  - <LRMSname>.pm: old style LRMS interface
  - <LRMSname>mod.pm
- ARC0mod.pm takes care of loading them seamlessly to return the correct values to LRMSinfo.pm – it bridges legacy LRMS modules from old infoproviders

Specific LRMS modules

- Differences in short:
  - <LRMSname>mod.pm is GLUE2 aware and should return information in more structured way
  - It adds nodes status (i.e. detailed information about nodes in a queue)
  - It is more compliant to PERL module style coding

Modules: what they do

- Information is divided in four blocks:
  - Cluster
  - Queues
  - Jobs
  - Nodes
- LRMS specific commands are run accordingly
- Some LRMS modules like Condor even read some files in the controldir
- Nothing is written in the controldir, everything is stored in memory and returned to LRMSinfo.pm

LRMSes present in SVN

- Boinc.pm
  Berkeley Volunteer computing project
- Condor.pm, condor_env.pm
  Condor/HTCondor
- DGBridge.pm
  Desktop Grids
- Fork.pm, FORKmod.pm
  ARC simple unix fork based submission system
- LL.pm
  Load Leveller
- LSF.pm
  IBM Load Sharing Facility
- PBS.pm
  PBS/Torque
- SGE.pm, SGEmod.pm
  Sun Grid Engine (today Oracle GE)
- SLURM.pm, SLURMmod.pm
  SchedMD Simple Linux Workload Management

Commands run: Boinc

```
DBI->connect("DBI:mysql:$DB_NAME;host=$DB_HOST:$DB_PORT","$DB_USER","$DB_PASS",
{RaiseError=>1});
```

totalcpus

```
$dbh->prepare('select sum(p_ncpus) from host where expavg_credit>0');
```

maxcpus

```
$dbh->prepare('select sum(p_ncpus) from host');
```

jobs_in_que

```
$dbh->prepare('select count(*) from result where server_state in (1,2)');
```

jobs_in_run

```
$dbh->prepare('select count(*) from result where server_state=4');
```

job_status

```
$dbh->prepare('select server_state,name from result where server_state in (1,2,4)');
```

Commands run: Condor

Condor

collect_node_data

```
condor_run('condor_status -format "Name = %V\n" Name -format "Machine = %V\n" Machine -format "State = %V\n" State -format "Cpus = %V\n" Cpus -format "TotalCpus = %V\n" TotalCpus -format "SlotType = %V\n\n" SlotType');
```

collect_job_data

```
condor_run('condor_q -constraint "NiceUser == False" -format "ClusterId = %V\n" ClusterId -format "ProcId = %V\n" ProcId -format "JobStatus = %V\n" JobStatus -format "CurrentHosts = %V\n" CurrentHosts -format "LastRemoteHost = %V\n" LastRemoteHost -format "RemoteHost = %V\n" RemoteHost -format "ImageSize = %V\n" ImageSize -format "RemoteWallClockTime = %V\n" RemoteWallClockTime -format "RemoteUserCpu = %V\n" RemoteUserCpu -format "RemoteSysCpu = %V\n" RemoteSysCpu -format "JobTimeLimit = %V\n" JobTimeLimit -format "JobCpuLimit = %V\n\n" JobCpuLimit');
```

collect_job_ids

```
my $cmd = "find $controldir/processing -maxdepth 1 -name 'job.??????????*.status'";
$cmd   .= ' | xargs grep -l INLRMS ';
$cmd   .= ' | sed \'s/processing\/job\.\([^\.]*\)\.status$/job.\1.local/\' ';
$cmd   .= ' | xargs grep -H "^queue=\|^localid="';
```

condor_grep_nodes

```
my $cmd = 'condor_status -format "%s\n" Machine';
```

Commands run: DGBridge

getDGstate

```
wsclient -e $ep -m status -j $jid |
```

Commands run: Fork

uses SysInfo.pm

process_info
ps -e -o ppid,pid,vsz,**time**,etime,user,**comm**

cluster_info
uptime

queue_info
ulimit -t

Commands run: LL

```
consumable_distribution
   unless (open LLSTATUSOUT,  "$path/llstatus -R|") {
get_cpu_distribution
   # Without hyperthreading
   unless (open LLSTATUSOUT,  "$path/llstatus -f %sta|") {
     # Use all available cpus/cores including hyperthreading:
  unless (open LLSTATUSOUT,  "$path/llstatus -r %cpu %sta|") {
get_used_cpus
  unless (open LLSTATUSOUT,  "$path/llstatus |") {
get_long_status
  unless (open LLSTATUSOUT,  "$path/llstatus -l |") {
get_long_queue_info
  unless (open LLCLASSOUT,  "$path/llclass -l $queue |") {
get_queues
  unless (open LLCLASSOUT,  "$path/llclass |") {
get_short_job_info
   unless (open LLQOUT, "$path/llq -c $queue |") {
   unless (open LLQOUT, "$path/llq |") {
get_long_job_info
  unless (open LLQOUT, "$path/llq -l -x $lrmsidstr |") {
cluster_info
   my $status_string=`$path/llstatus -v`;
```

Commands run: LSF

```
$lshosts_command="$path/lshosts -w";
$bhosts_command = "$path/bhosts -w";
$bqueues_command = "$path/bqueues -w";
$bqueuesl_command = "$path/bqueues -l";
$bjobs_command = "$path/bjobs -W -w";
$lsid_command="$path/lsid";
read_lsfnodes
   unless (open LSFHOSTSOUTPUT,   "$lshosts_command |") {
   unless (open LSFBHOSTSOUTPUT, "$bhosts_command |") {
type_and_version
    my (@lsf_version) = `$lsid_command -V 2>&1`;
queue_info_user
   unless ($user eq ""){
     $user = "-u " . $user;
   }
   unless (open BQOUTPUT,  "$bqueues_command $user $qname|") {
   unless (open BQOUTPUT,  "$bqueuesl_command $user $qname|") {
get_jobinfo
  unless (open BJOUTPUT,  "$bjobs_command $id|") {
cluster_info
   unless (open BQOUTPUT,   "$bqueues_command|") {
```

Commands run: PBS

```
get_pbs_version
    my $qmgr_string=`$path/qmgr -c "list server"`;
read_pbsnodes
    unless (open PBSNODESOUT,  "$path/pbsnodes -a 2>/dev/null |") {
read_qstat_f
    unless ( open QSTAT_F, "$path/qstat -f 2>/dev/null |") {
process_dqueues
     unless (open QSTATOUTPUT,   "$path/qstat -Q -f $dqname 2>/dev/null |") {
cluster_info
    unless (open QSTATOUTPUT,  "$path/qstat -Q 2>/dev/null |") {
queue_info
    unless (open QSTATOUTPUT,   "$path/qstat -Q -f $qname 2>/dev/null |") {
    unless (open QSTATOUTPUT,   "$path/qstat -Q -f $qname 2>/dev/null |") {
jobs_info
 my $showq = (defined $$config{maui_bin_path}) ? $$config{maui_bin_path}."/showq" : "showq";
    unless (open SHOWQOUTPUT, " $showq |"){
   unless (open QSTATOUTPUT,   "$path/qstat -f 2>/dev/null |") {
users_info
    unless (open QSTATOUTPUT,   "$path/qstat -f -Q $qname 2>/dev/null |") {
    unless (open QSTATOUTPUT,   "$path/qstat -f -Q $singledqueue 2>/dev/null |") {
    my $showbf = (defined $$config{maui_bin_path}) ? $$config{maui_bin_path}."/showbf" : "showbf";
    unless (open SHOWBFOUTPUT, " $showbf -u $u |"){
```

Commands run: SGE

type_and_version

```
run_callback("$path/qstat -help", sub {
```

run_qstat

```
my $command = "$path/qstat -u '*'";
$command .= $compat_mode ? " -F" : " -f";
```

run_qconf

```
my $qhost_xml_output = `$path/qhost -xml` or $log->error("Failed listing licensed processors");
# global limits
loop_callback("$path/qconf -sconf global", sub {
# list all queues
loop_callback("$path/qconf -sql", sub {
```

queue_info

```
my $command = "$path/qconf -sq @qnames";
```

jobs_info

```
# Running jobs
loop_callback("$path/qstat -j $jidstr", sub {
# Waiting jobs
loop_callback("$path/qstat -j $jidstr", sub {
```

run_qhost

```
loop_callback("$path/qhost -F -h `echo $host | cut -d . -f 1` | grep '='", sub {
```

check_host_state_na

```
loop_callback("$path/qstat -f | grep `echo $host | cut -d . -f 1`", sub {
```

Commands run: SLURM

```
slurm_read_config
   open (SCPIPE,"$path/scontrol show config| grep -Ev \"primary|Configuration|^\$\"|");
slurm_read_partitions
   open (SCPIPE,"$path/sinfo -a -h -o \"PartitionName=%P TotalCPUs=%C TotalNodes=%D MaxTime=%l\"|");
slurm_read_jobs
   open (SCPIPE,"$path/squeue -a -h -t all -o \"JobId=%i TimeUsed=%M Partition=%P JobState=%T ReqNodes=%D ReqCPUs=%C TimeLimit=%l Name=%j NodeList=%N\"|");
slurm_read_nodes
   open (SCPIPE,"$path/scontrol show node –oneliner|");
slurm_read_cpuinfo
   open (SCPIPE,"$path/sinfo -a -h -o \"cpuinfo=%C\"|");
```