Resource Utilization Reporting

CUG 2013
Andrew Barry, Cray Inc.
Warnings About RUR

- RUR is unreleased software
- Not all features discussed will be available in the first release of RUR
Why RUR?

- Cray administrators want to know more about how systems are being used.
- This may impact future procurements, administration decisions, or be used for billing.
- Cray has previously offered several accounting tools, which were compelling at the time, but don’t meet current needs.
  - **CSA** – designed for single system image, X2 port usable with lustre storage; lustre performance impact with large node counts.
  - **Mazama ACR** – complex infrastructure and database tuning, to support lone remaining Mazama feature.
  - **ARU** – limited release software, stop-gap to RUR Limited customization.
Why RUR?

- Third Party Tools don't solve the right problem
- Server Management Tools:
  - Nagios/Shinken/Incincga, Ganglia, PandoraFMS, OpenNMS, NetXMS
- Server Managers collect real-time data, not application-scale data
- OS-Noise, timescale granularity concerns
- Batch Scheduler Accounting: PBS, Moab, Slurm, LSF
- Need MOM on compute nodes, limited data
What is RUR?

- Tool for collecting statistics about how system resources are used by applications
  - Scalable
  - extensible
  - configurable
  - lightweight
- Plugins architecture
  - collection of arbitrary data from compute nodes
  - post-processing
  - storing data anywhere
  - integration with existing tools
- CLE 4.2up02 and 5.1up00
RUR Components

- **Data staging** on the compute nodes
- **Data collection** from computes to login
- **post-processing** to summarize data
- **logging/storage** of the output
RUR

Login Node
- Aprun
- Gather
- Post-Process
- Output

Compute Nodes
- Launch
- ExitMsg
- Fanout
- Apinit
- Staging
- Application

Summary Data
- Log File
- LLM

Alps
- RUR

SMW
RUR Plugin Support

- RUR is extensible through plugins
- Plugins will each have a staging, and post-processing, and may have a logging component
- RUR infrastructure will make it simple to write basic plugins
- Can collect any data available on the target nodes
- Post-processing can be arbitrarily complex
- Store to any log, database, etc
RUR

Login Node

- Aprun
- Gather
- Post-Process
  - Plugins
- Output
  - Plugins

Compute Nodes

- Launch
- ExitMsg
- Gather
- Apinit
- Application
- Staging
- Plugins

SMW

Log File

Alternate Backing Store

LLM

Summary Data

SQL
RUR Phases

● **Data Staging on compute nodes in two phases**
  ● Collect data before the application run
  ● Collect data after the application run
  ● The staged data is the delta of the two
  ● Plugin-specific

● **Data collection:** After application is run, rur-gather on the login/mom node gathers data from compute nodes. In the future can also be launched in batch epilogue

● **The data collection uses a resilient fanout tree, with a timeout**

● **Post-processing:** built-in support for sum, min, max, mean, and histogram operations
RUR and Process Accounting

- Cray developed RUR plugin
- Rusage / BSD acct style accounting
- CPU utime, stime, Memory Highwater, File I/O
- **Pre-app**: Clear taskstats buffer
- **Post-app**: Collect taskstats buffer
- **Post-processing**: CPU: sum, MemoryHighwater: max, FileI/O: sum
- **Data storage**: Log file with LLM
- **Record includes**: userid, apid, jobid, cmdname, aggregated data
RUR and GPU Accounting

- Cray developed RUR plugin
- Cray GPU accounting utility on compute nodes interface to NVML
- **Pre-app:** Zero the GPU counters
- **Post-app:** Collect GPU counters
- **Post-processing:** MemoryUsed: max, TimeUsed: sum, Utilization: mean
- **Data storage:** Log file with LLM
- **Record includes:** userid, apid, jobid, cmdname, aggregated data
Cray developed RUR plugin

Track power usage on compute nodes for application duration

**Pre-app:** Collect node initial joules used

**Post-app:** Collect node final joules used

**Post-processing:** Sum all energy used on all nodes for the application duration

**Data storage:** Log file with LLM

**Record includes:** Userid, apid, jobid, cmdname, aggregated data
RUR and Site-Custom Plugins

- Admin developed RUR plugin.
- Collect data from “widget” software, running on compute nodes
- **Pre-app:** Collect initial number of widgets served
- **Post-app:** Collect post-app number of widgets served
- **Post-processing:** Sum of widgets served across all nodes, for application duration
- **Data storage:** Log file with LLM
RUR and Data-Rich Custom Plugins

- Admin developed RUR plugin
- Collect data from advanced widget server, running on compute nodes
- **Pre-app:** Collect initial widgets statistic matrix
- **Post-app:** Collect post-app widgets statistic matrix
- **Post-processing:** Custom generated histogram of many widget statistics
- **Data storage:** Custom output plugin records histogram in widget statistic database table; RUR LLM log file includes an record index
Possible Future RUR plugins

- Minor errors
- Lustre Filesystem statistics
- DVS statistics
- Per-mountpoint statistics
- Aries performance counters
- Future coprocessors
- Error codes / application completion
Possible RUR Output Plugins

- Existing database
- System visualization tool
- Interface to WLM accounting
- Output to user in batch output
- Email to admin
Job Scale Reporting

- Initially, job-scale reporting will be the sum of the applications in the job.
- A user may reserve more nodes than the application actually runs on.
- RUR components can be initiated by the WLM, rather than ALPS, to provide true job-scale data.
Questions?