Monitoring and Analyzing Job Performance Using Resource Utilization Reporting (RUR) on A Cray XE6/XK6 System

Shi-Quan Su, Troy Baer, Gary Rogers, Stephen McNally,
Robert Whitten, Lonnie Crosby,

National Institute for Computational Sciences

University of Tennessee, USA;

Research sponsors: the U.S. National Science Foundation

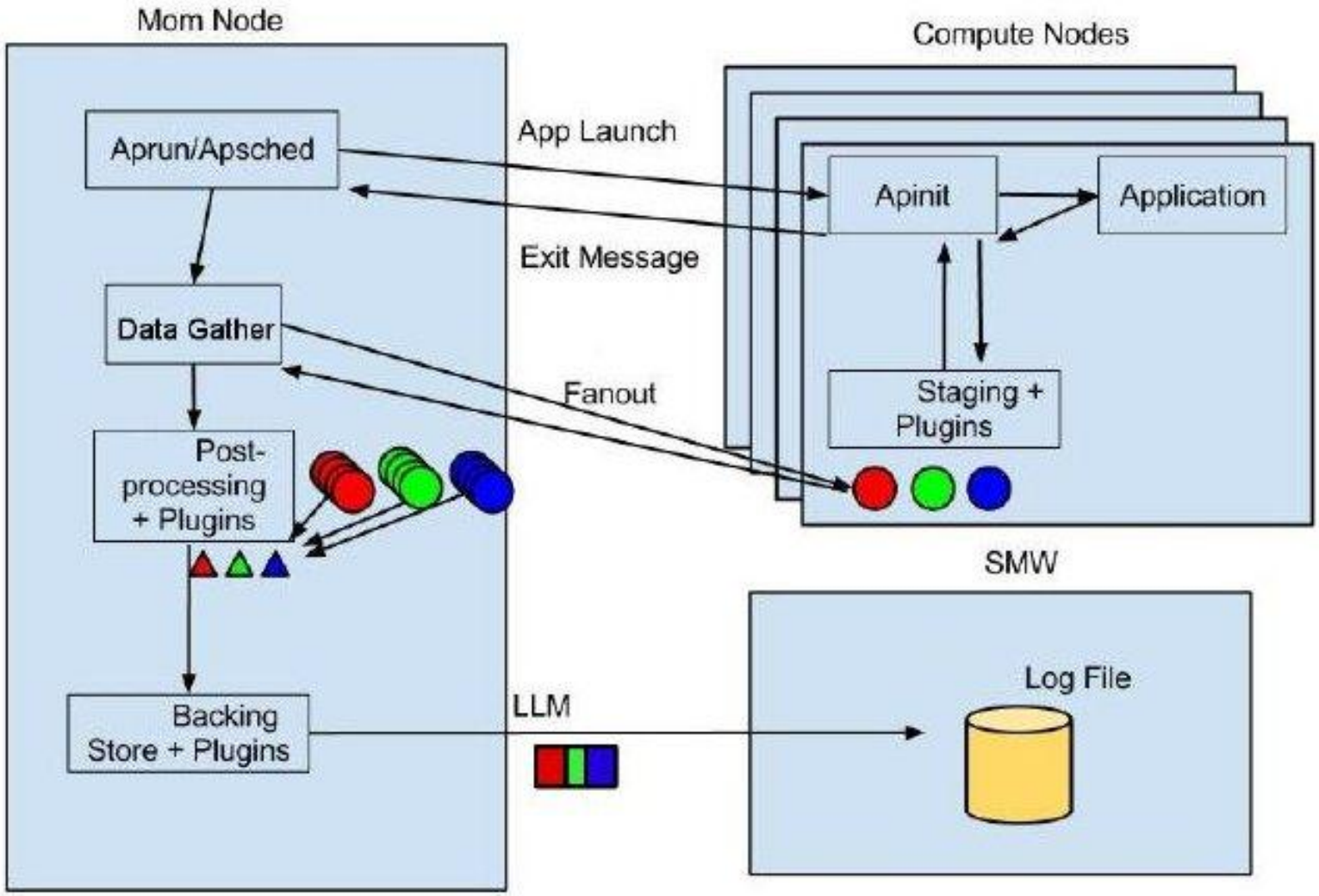
CUG Conference 2015

Outline

- Brief Review of RUR
- Quantifying RUR Overhead
- Integration with Torque
- Generating Single Job Record
- Integration with XDMoD
- RUR Configuration Experiences
- Discussions and Conclusions

Brief Review of RUR

RUR provides a reliable, high-performance framework into which plugins may be inserted, which will collect data about the usage of a particular resource. RUR is configurable, extensible, and lightweight.



Source: Resource Utilization Reporting Gathering and evaluating HPC system usage, Andrew Barry Cray Inc.

Quantifying RUR Overhead

Test Platform:

Mars at NICS, a Cray XE6/XK6 system, running Moab and Torque.

Test Harness:

The Test Harness platform is originally developed at ORNL, and has been used in the acceptance test on several Cray systems such as Jaguar, Kraken, Darter, and Titan. The Test Harness includes two major components：

1, Test Harness Python library,

2, the set of benchmarks of various applications.

Running Test Harness

1, the Test Harness initially compiles and links the applications,

2, submits the benchmark runs to the batch system,

3, collects the job outputs, and compares the results against the built in test,

4, the Test Harness records the results locally and archives the job data into the High Performance Storage System (HPSS) at Oak Ridge National Laboratory (ORNL) campus,

5, the Test Harness automatically restarts the process until a predetermined time passes, or a predetermined number of runs have completed.

Structure of Test Harness

xc30/NAMD/Source/"Source code"
---------- /input_files/apoa1
----------/dhfr
----------/stmv
----------/build_app.sh
----------/[TestCase]:apoa1_32/Correct_Results/ #same for each [TestCase]
----------/[TestCase]:apoa1_32/Scripts/archive.template.x #same for each [TestCase]
----------/build_executable.x #same for each [TestCase]
----------/check_executable.x #same for each [TestCase]
----------/pbs.template.x #same for each [TestCase]
----------/submit_executable.x #** use job_tag to distingush TestCase
----------/size.txt #** may set different # of processor
----------/RunArchive/[unique_id]/"results directory"
----------/Status/rgt_status.txt
----------/[unique_id]/job_id.txt #** different for each [unique_id]
----------/job_status.txt # either 0 (success) 1(fail) 2(not clear)
----------/[TestCase]:dhfr_64/Correct_Results/ #same for each [TestCase]
----------/[TestCase]:dhfr_64/Scripts/archive.template.x #same for each [TestCase]
----------/build_executable.x #same for each [TestCase]
----------/check_executable.x #same for each [TestCase]
----------/pbs.template.x #same for each [TestCase]
----------/submit_executable.x #** use job_tag to distingush TestCase
----------/size.txt #** may set different # of processor
----------/RunArchive/[unique_id]/"results directory"
----------/Status/rgt_status.txt
----------/[unique_id]/job_id.txt #** different for each [unique_id]
----------/job_status.txt # either 0 (success) 1(fail) 2(not clear)

Test Harness Applications and Benchmarks

NAMD: a parallel molecular dynamics code designed for high-performance simulation of large bio-molecular systems.

benchmark: apoa1, stmv

NWCHEM: a scalable computational chemistry package, ackling molecular systems including biomolecules, nanostructures, actinide complexes, and materials.

benchmark: aump2, c60_pbe0

GROMACS: a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles.

benchmark: d.dppc, d.villin

TABLE OF TEST HARNESS BENCHMARKS PERFORMANCE

Benchmark	Cores	RUR	Jobs	Ave. Run Time	Std. Dev.
stmv	128	on	388	129.71	0.37
	128	off	398	129.68	0.45
stmv	256	on	227	89.73	1.47
	256	off	240	88.49	1.29
apoa1	32	on	1102	81.67	0.42
	32	off	1261	81.86	0.48
apoa1	64	on	1385	46.21	0.36
	64	off	1519	46.25	0.78
c60_pbe0	32	on	60	322.81	2.69
	32	off	52	322.60	2.88
c60_pbe0	64	on	98	167.45	1.21
	64	off	80	167.64	1.38
aump2	32	on	56	349.77	2.71
	32	off	60	349.67	2.05
aump2	64	on	89	184.92	1.41
	64	off	75	185.50	2.05
d.dppc	64	on	71	114.02	0.16
	64	off	55	114.05	0.16
d.dppc	128	on	96	64.68	0.16
	128	off	77	64.77	0.18
d.villin	32	on	109	74.22	0.15
	32	off	86	74.26	0.13
d.villin	64	on	134	44.85	0.37
	64	off	104	44.92	0.35

Integration with Torque

- TORQUE version 5.0 added a field in its accounting logs for recoding the energy used by a job, called resources used.energy used. The initial implementation of this can be found in src/resmom/cray energy.c in the source code. This implementation targets RUR, specifically the energy plugin, as its underlying data gathering infrastructure.
- The RUR energy plugin does not support energy measurement on the XE6/XK6 hardware platform, but that fact was not documented in the Cray system software manual describing RUR.
- NICS has requested that Cray update the documentation.

Generating Single Job Record:

rchar, wchar,
stime, utime,
exitcode, max_rss,
%_of_boot_mem,
Active(anon), Active(file),
gid, jid, nid, uid, apid,
APP_START, APP_STOP

RUR data

system name, job id,
user name, group name,
charge account, job name,
number of processers requested,
queue name,
submit time, start time,
end time, wall time requested,
allocated host list, exit status,
node where the job was submitted,
user's batch job script.

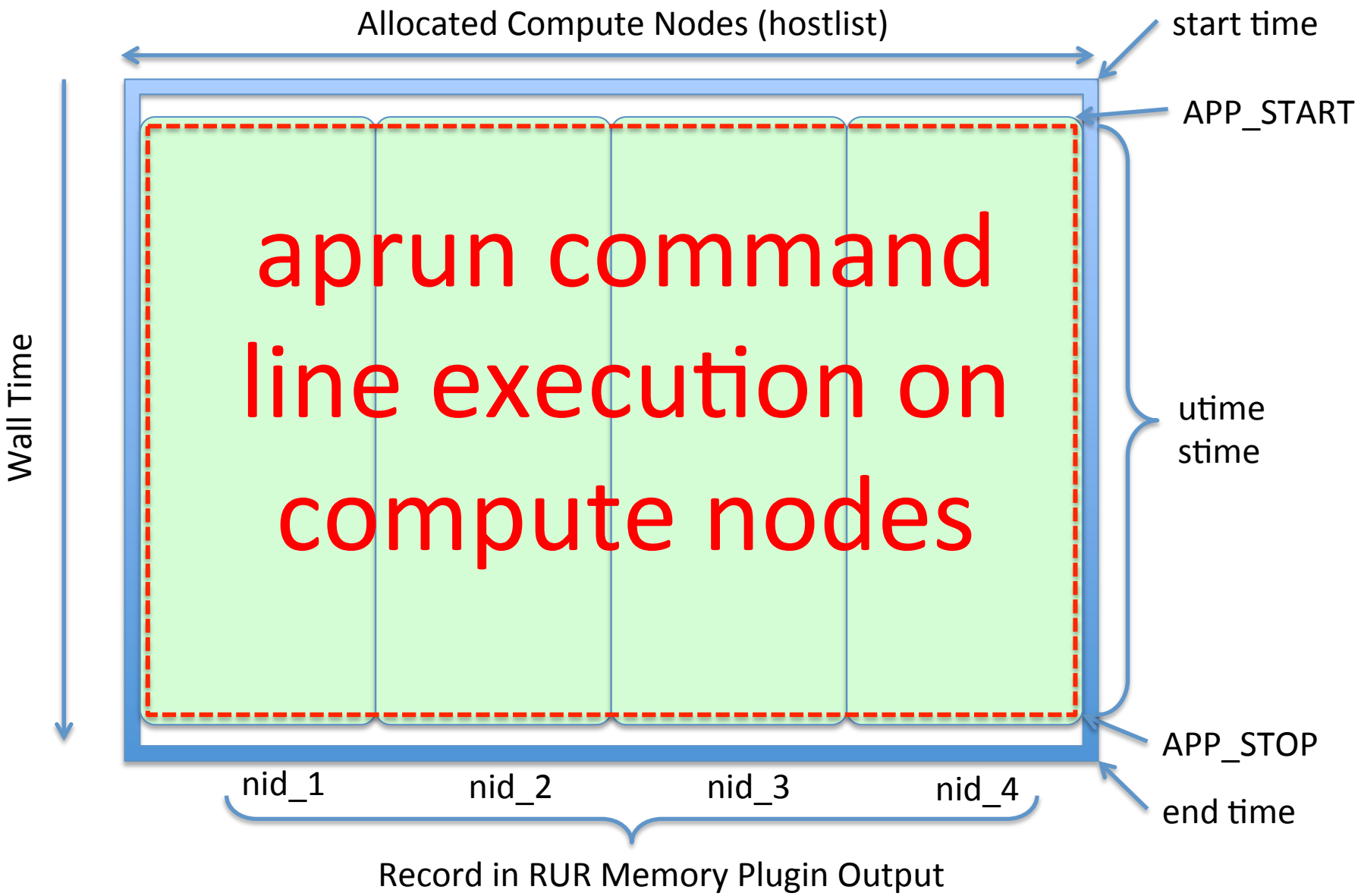
pbsacct utility

After parsing the RUR output and querying the pbsacct database, a single job record can be generated in a json dictionary style. Each job record is stored in a file. These records can be injected into external analysis systems for further processing, such as XDMoD.

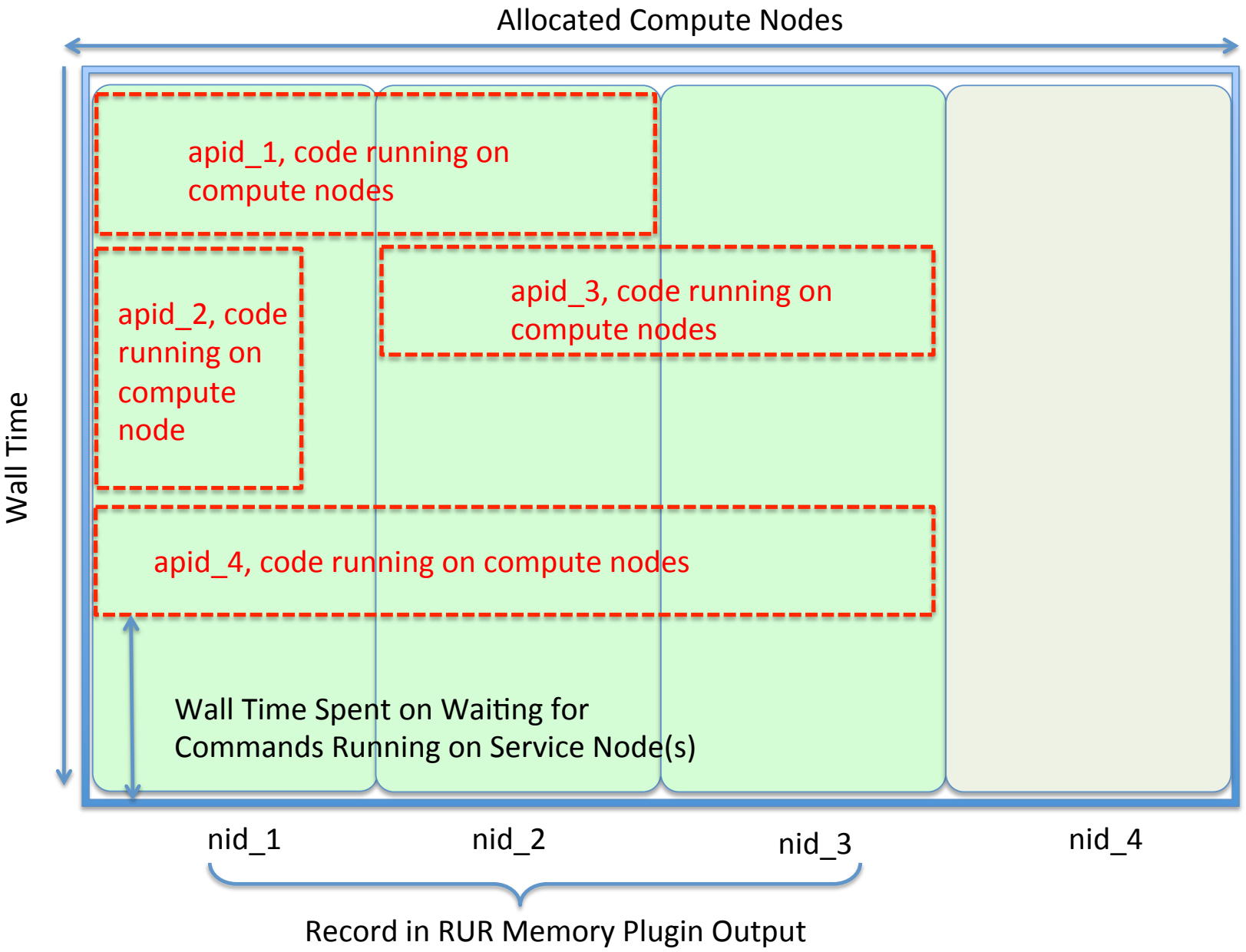
Integration with XDMoD

- The XSEDE Technology Audit Service (TAS) is an NSF funded project at the Center for Computational Research (CCR) of the State University of New York (SUNY) at Buffalo to audit the XSEDE project's activities.
- One of this project's main products is XSEDE Metrics on Demand (XDMoD), a software system which provides web-based access to metrics and analytics for the computational resources of the XSEDE service providers (SPs).
- The XDMoD (https: //xdmod.ccr.buffalo.edu) is designed to audit and facilitate the operation and utilization of XSEDE, one of the most advanced and robust collection of integrated advanced digital resources and services in the world.

A schematic show of a simple job structure



A schematic show of a complex job structure



“cpu idle percentage” (ρ)

$$\rho = \frac{\sum_{i=1}^{n} x_i}{n}$$

where n is number of cores the job ran on. x_i is cpu idle percentage of each core.

$$\rho = 1 - \frac{\sum_{i=1}^{k} (utime_i + stime_i)}{T \times N}$$

where k is the number of aprun, $utime_i$ and $stime_i$ are from the RUR taskstats plugin record with apid $_i$, T is the wall time elapsed for the whole job, N is the number of cores actually assigned to the job, sum up the values of ppn_{nid} with the nid in array of allocated host names from job accounting information

”memory usage including system service per node” (μ)

$$\mu = \frac{\sum_{i=1}^{H} \frac{x_i}{C_i}}{H}$$

where x_i is the mean memory used on node i, C_i is the number of cores on node i and H is the number of nodes on which the job ran.

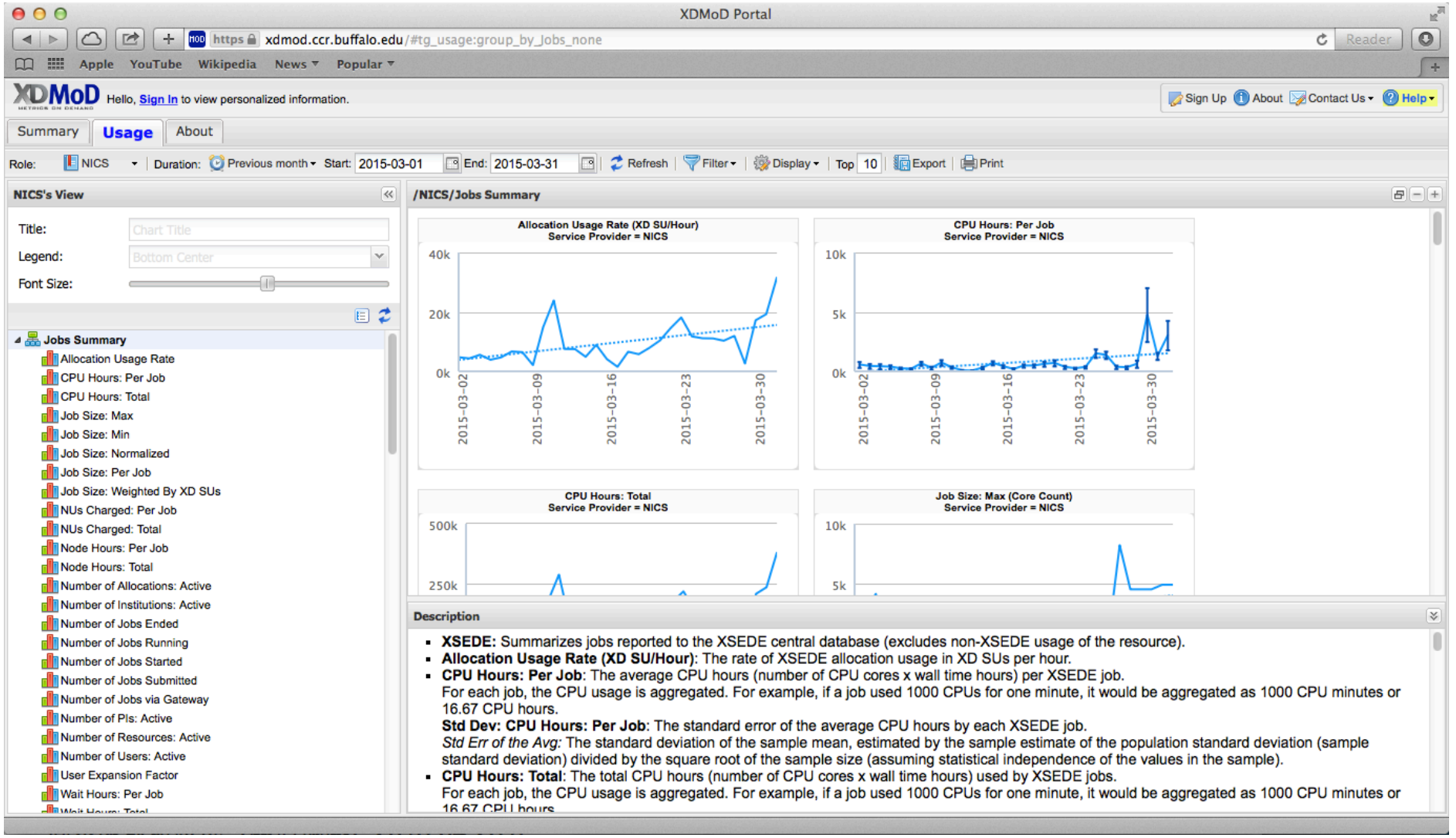
$$\mu = \sum_{j=1}^{k} w_j \times \frac{\sum_{i=1}^{n_j} m_i}{n_j}$$

where k is the number of aprun, n_j is the number of nodes in the jth aprun, m_i is the ”Active(anon)” value in the RUR memory plugin output with $apid_j$, and nid_i. The w_i is the weight calculated as below:

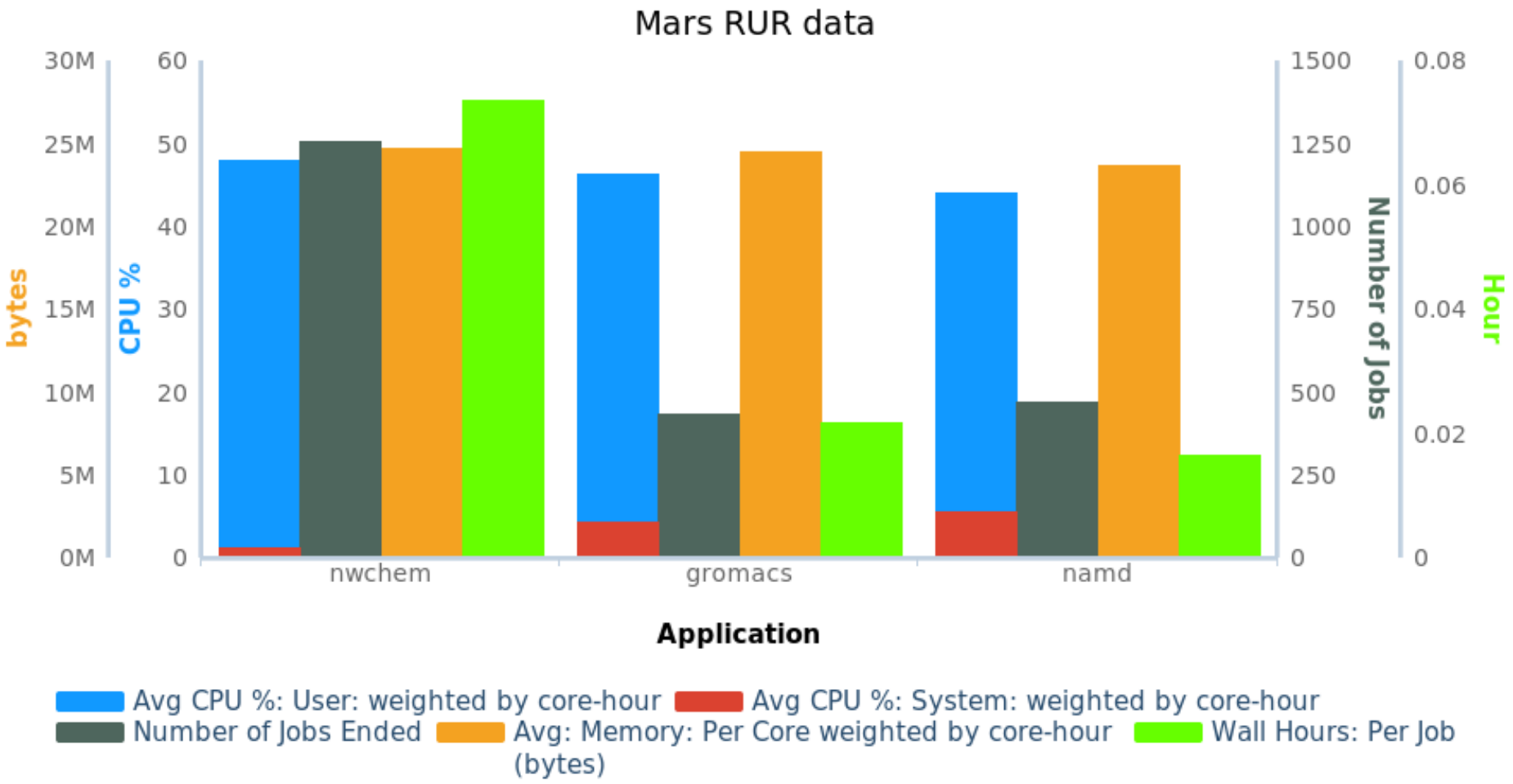
$$w_j = \frac{(APP\_STOP_j - APP\_START_j) * N_j)}{T \times N}$$

where APP_STOP_j and APP_START_j are values from RUR timestamp plugin with $apid_j$.

XDMoD Website Interface



Example of analyzing RUR data via XDMoD



RUR Configuration Experiences

alps.conf

```
apsys
  prologPath /opt/cray/rur/default/bin/rur_prologue.py
  epilogPath /opt/cray/rur/default/bin/rur_epilogue.py
  prologTimeout 60
  epilogTimeout 60
/apsys
```

rur.conf

```
[global]
rur: False

[rur_stage]
stage_timeout: 10
stage_dir: /tmp/rur/

[rur_gather]
gather_timeout: 10
gather_dir: /tmp/rur/

[rur_post]
post_timeout: 10
post_dir: /tmp/rur/

[plugins]
gpustat: false
taskstats: true
timestamp: true
energy: false
memory: true

[outputplugins]
llm: false
file: true
user: false
```

```
[gpustat]
stage:  /opt/cray/rur/default/bin/gpustat_stage.py
post:   /opt/cray/rur/default/bin/gpustat_post.py

[taskstats]
stage:  /opt/cray/rur/default/bin/taskstats_stage.py
post:   /opt/cray/rur/default/bin/taskstats_post.py

[energy]
stage:  /opt/cray/rur/default/bin/energy_stage.py
post:   /opt/cray/rur/default/bin/energy_post.py

[timestamp]
stage:  /opt/cray/rur/default/bin/timestamp_stage.py
post:   /opt/cray/rur/default/bin/timestamp_post.py

[memory]
stage:  /opt/cray/rur/default/bin/memory_stage.py
post:   /opt/cray/rur/default/bin/memory_post.py


[llm]
output: /opt/cray/rur/default/bin/llm_output.py

[file]
output: /opt/cray/rur/default/bin/file_output.py
arg:    /lustre/medusa/grogers/RUR/output/rur.output

[user]
output: /opt/cray/rur/default/bin/user_output.py
arg:    single
```

Discussions and Conclusions

- The RUR provides a low noise, scalable approach to collect performance data from the compute nodes.
- The RUR framework allows users develop customized plugins for various purposes.
- It appears that RUR is a stable utility going forward serving the Cray community. There are substantial development works from multiple aspects related to RUR.

Discussions and Conclusions

- The data generated by the default plugins show that RUR, in its current state, is at best a supplement for the accounting data recorded by whatever resource management system the site implements.
- The existing plugins can give additional data on CPU time, memory, energy, and GPU usage on the compute nodes that is not typically available in the resource manager accounting logs on Cray systems.

Conclusions and Discussions

- However, most of the deeper performance information desired by third-party analytics systems such as XDMoD and provided by other monitoring systems such as TACC Stats are not available through RUR. Specifically, plugins for processor performance counters, interconnect performance counters, and file system performance counters (such as for Lustre) are not currently available for RUR on XE6/XK6 platforms.

THANK YOU