RRZ Batch Job Reports

Thomas Orgis, Hinnerk Stüben

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

— Regionales Rechenzentrum (RRZ) —

ZKI-Arbeitskreis Supercomputing

7 October 2021

Overview

Introduction

- motivation
  - improve resource utilisation

  $\rightarrow$ provide resource usage information to users and admins
- history
  - since 2015
    - predecessors
    - internal use
  - since January 2020
    - recommended to individual users
  - August 2021
    - user documentation written
    - general availability announced to users

Which data is collected and reported?

- CPU (per hyperthread)
  - user, sys, iowait, idle
- memory (per node)
  - high-water marks of RSS and VM
- disk I/O (per disk system)
  - amounts of data being read/written, bandwidth, IOPS
- communication network (per node)
  - bandwidth
- GPU (per device)
  - power load, GPU load
  - memory: load, use
- processes that used most resources
  - number of invocations, multi-threading

Data sources

- CPU: `/proc/stat`
- memory: `/proc/meminfo`
- disk I/O:
  - local: `/proc/diskstats`
  - NFS: `/proc/self/mountstats`
  - BeeGFS: `beegfs-ctl`
- GPU: `nvidia-smi`
- network:
  - ethernet: `/proc/net/dev`
  - infiniband: `perfquery -xa`
- per process/task statistics:
  - taskstats netlink kernel interface (exact high-water marks of RSS and VM)

Design goals

- like LIKWID* we "like to know what we are doing"
- light weight (minimal software dependences, no containers)
- simplicity (avoid time series)
- locality (no central component like a data bank)
- orthogonality (independence of the batch system)
- minimal privileges (no *suid* programs, *root* only where unavoidable)

* *'LIKWID stands for "Like I Knew What I'm Doing."'*

→ `https://hpc.fau.de/research/tools/likwid/`

Implementation

- script suite written in Perl (≈ 5000 lines of code)
- privileges
  - monitoring as unprivileged user
  - most files/programs are public resources
  - `perfquery` is called from a script contacting a local privilege separation daemon that can *only* call `perfquery -a` or `perfquery -xa`
  - `nvidia-smi` can be called by users
    `nvidia-modprobe` is run by root at boot time
- front-ends
  - `rrz-batch-jobreport` – for completed batch jobs
  - `rrz-batch-use` – for running batch jobs

Structure/sections of a report

- Header
  - job ID, type of CPU (and GPU), elapsed time
- Data amounts
  - virtual memory, disk I/O, data communication
- Per node resource usage
  - CPU load, memory and swap high-water marks, disk IOPS and bandwidths, data communication bandwidth
- Per core resource usage (multi-node jobs: averages over nodes)
  - CPU utilisation per physical core
  - GPU jobs: GPU load, GPU memory load
- Per command resource usage
  - commands that used most resources (including amounts)
- Summary
  - quick overview

Job report summary

- example (GPU information appears only for GPU jobs)

```
Summary:

    Elapsed time: 11%  (0.0 out of 0.2 h timelimit)
             GPU: 78%  (1.6 out of 2 GPUs)
             CPU: 12%  (1.9 out of 16 physical CPU cores)
    Hyperthreads:  0%  (0.0 out of 16 CPU hyperthreads)
 Max. GPU memory: 90% (10.1 out of 11.2 GiB per GPU)
Max. main memory:  2%  (1.5 out of 62.0 GiB min. available per node)
       Max. swap:  0%  (0.0 out of  2.0 GiB min. available per node)
```

- implies hints for action

What can be learned from job reports?

- time limit: can it be lowered? (to improve backfill scheduling)
- CPU/GPU: are there unused nodes/cores/GPUs? is utilisation high enough?
- CPU/GPU: is load balanced?
- memory usage: can smaller nodes be used?
- I/O: are there hints of an I/O bottleneck?
  - are data amounts large / much higher than expected from file sizes?
  - is the average I/O bandwidth large?

Documentation

- front-ends

  `https://www.rrz.uni-hamburg.de/en/services/hpc/hummel-2015/rrz-tools/...`

  `rrz-batch-jobreport`

  `rrz-batch-use`

- example job reports

  `https://www.rrz.uni-hamburg.de/en/services/hpc/hummel-2015/rrz-tools/...`

  `rrz-batch-jobreport/single-node-report`

  `rrz-batch-jobreport/multi-node-report`

  `rrz-batch-jobreport/gpu-job-report`

→ demo: example job reports

Conclusion

- achievement
  - batch job reports help to understand and improve HPC-cluster utilisation
- wish list
  - automatic detection of jobs that should be improved
  - inclusion of performance figures (e.g. from LIKWID)