Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

Monitoring jobs resource usage in real-time

Bastien Gounon

January 27, 2020

CCIN2P3

cnrs

Goal

- Understand resource usage for LSST jobs running at CC-IN2P3
  - CPU efficiency, peak memory usage, reads and writes, ...
- Grid Engine accounting ?
  - unreliable I/O counters...
  - stats only available after job completion
- Need for a new, more flexible, lightweight profiling tool
  - provide real-time information
  - help with occasional stalled jobs: “what’s happening ?”
  - help with post-mortem of failed jobs
  - optimize resource allocation for each task

Linux procfs

- Pseudo-filesystem available on all Linux systems (and some more)
- Kernel-level, reliable source of information about running processes
- Contains detailed information about resource usage
  - /proc/[pid]/stat: process status, CPU time, allocated memory, ...
  - /proc/[pid]/io: read / written bytes
  - /proc/[pid]/task/[tid]/children: list of children processes
  - and much more

```
> ls /proc/1203
attr          comm             fd        loginuid   mountstats  oom_score_adj  schedstat     stat       timerslack_ns
autogroup     coredump_filter  fdinfo    map_files  net         pagemap        sessionid     statm      uid_map
auxv          cpuset           gid_map   maps       ns          personality    setgroups     status     wchan
cgroup        cwd              io        mem        numa_maps   projid_map     smaps         syscall
clear_refs    environ          latency   mountinfo  oom_adj     root           smaps_rollup  task
cmdline       exe              limits    mounts     oom_score   sched          stack         timers
```

Linux procfs

- /!\ Child process counters are not integrated into the parent’s until it exits

JoReMon: a metrics collection tool

- JoReMon is written in Go, based on prometheus/procfs
- Portable and standalone executable, very easy to deploy and run
- Also available as a Go library for developers
- Generates time-series of multiple metrics compiled from procfs
  - CPU: time and current usage
  - IO: bytes read and written on disk / in total
  - Memory: RSS (RAM) and VMS (RAM + swap)
- Configurable frequency
- Outputs parsable JSON or CSV file

JoReMon: how to use it ?

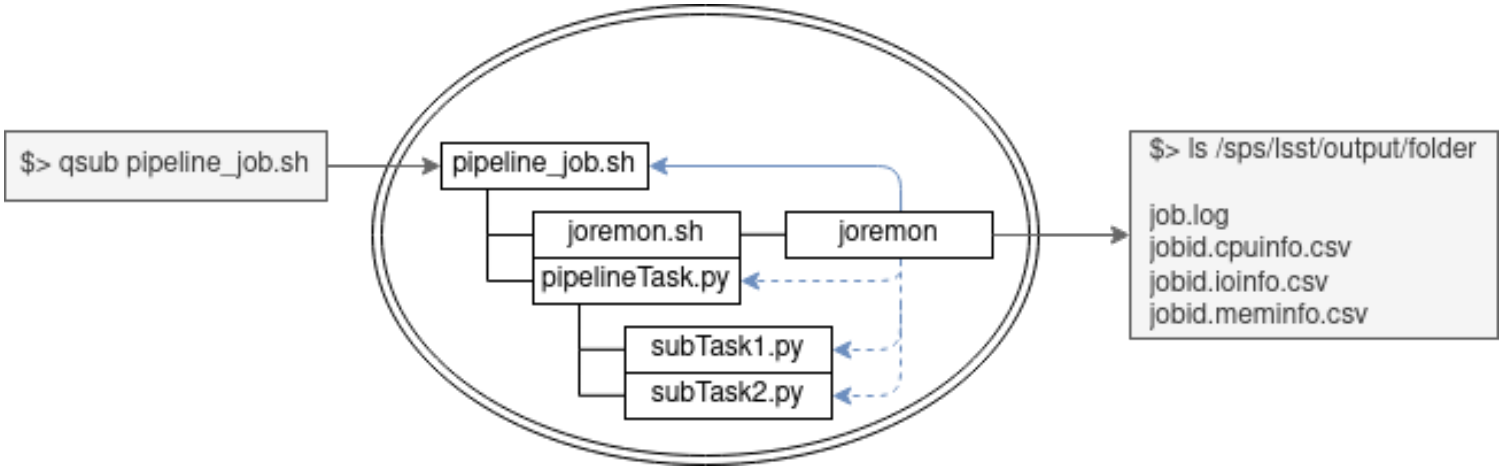
```
Usage of ./joremon:
  -folder string
        Output folder (ex: /tmp) (default ".")
  -format string
        Output format (csv / jsons) (default "jsons")
  -interval int
        Time interval between reports (default 5)
  -job string
        Job ID (will be reported in each row)
  -mode string
        Reporting mode (cpu / mem / io / all) (default "all")
  -out string
        Output file name (ex: myfile)
  -pid int
        Process ID (default 1)

Output values are:
- JobID : unique identifier
- Type : joremon.IOInfo / joremon.CPUInfo / joremon.MemInfo
- PID : which process is being watched
- CWD : current working directory
- CMD : which command is being run
- Hostname : name of the machine on which JoReMon is running
- User : who is running the process
- Timestamp : date and time at which the measure was taken
- CPUTime : cumulative CPU time used by the process
- ClockTime : time since process start
- CPUPercent : current CPU% usage
- PeakCPUPercent : maximum CPU% measured
- RBytesFromDisk : cumulative I/O read from disk, in bytes
- WBytesFromDisk : cumulative I/O written to disk, in bytes
- RBytesTotal : total cumulative I/O read, in bytes
- WBytesTotal : total cumulative I/O written, in bytes
- RSS : current main memory usage (RAM)
- PeakRSS : maximum main memory usage measured
- VMS : current virtual memory usage

More information at https://gitlab.in2p3.fr/bastien.gounon/joremon
```

```
> ./joremon -format jsons -interval 5 -job 123456 -mode cpu -pid 12889 -out -
{"JobID":"123456","Type":"joremon.CPUInfo","PID":12889,"CWD":"/pbs/home/d/descprod/bin","CMD":"-bash","Hostname":"cca002","User":"descprod","Timestamp":"2020-02-05T02:11:55.797358503+01:00","CPUTime":0.85,"ClockTime":187.470000028610 23,"CPUPercent":0.05294794049972205,"PeakCPUPercent":0.05294794049972205}
{"JobID":"123456","Type":"joremon.CPUInfo","PID":12889,"CWD":"/pbs/home/d/descprod/bin","CMD":"-bash","Hostname":"cca002","User":"descprod","Timestamp":"2020-02-05T02:12:00.864879228+01:00","CPUTime":0.85,"ClockTime":192.470000028610 23,"CPUPercent":0.05156453298031232,"PeakCPUPercent":0.05294794049972205}
```

```
> ./joremon -format csv -interval 5 -job 123456 -mode cpu -pid 12889 -out -
CMD,CPUPercent,CPUTime,CWD,ClockTime,Hostname,JobID,PID,PeakCPUPercent,Timestamp,Type,User
-bash,0.031922,1.010000,/pbs/home/d/descprod/bin,374.470000,cca002,123456,12889.000000,0.031922,2020-02-05T02:15:02.841672957+01:00,joremon.CPUInfo,descprod
-bash,0.031497,1.010000,/pbs/home/d/descprod/bin,379.470000,cca002,123456,12889.000000,0.031922,2020-02-05T02:15:07.917985941+01:00,joremon.CPUInfo,descprod
```

JoReMon: how do we use it ?

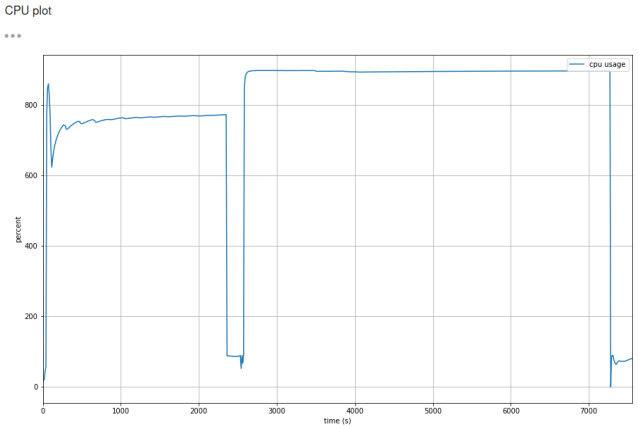
- Hide parameters and binary behind a wrapper
- Very easy to integrate into existing workloads
  - Call joremon.sh at the start of the job script
- Analog to the “sidecar” in container-based architectures

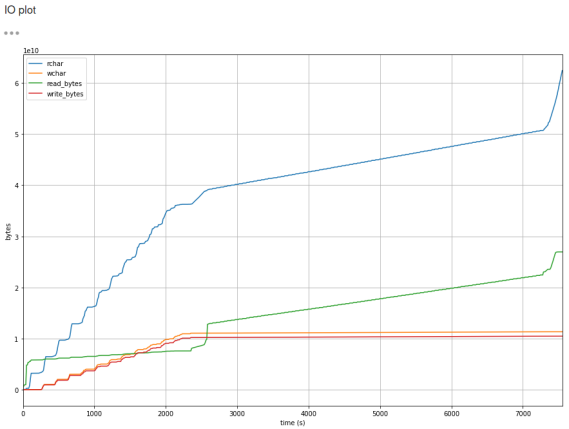


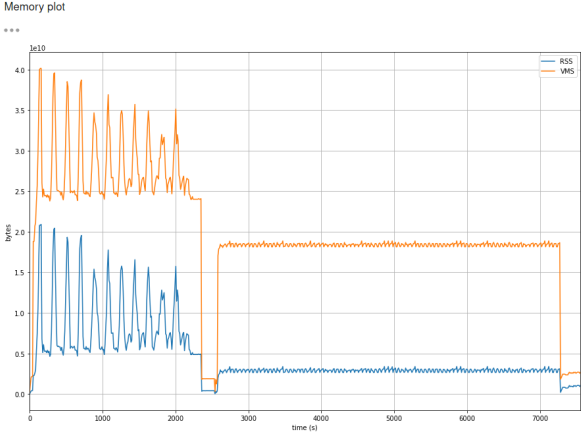
Visualization and analysis

- A collection of Jupyter notebooks to parse JoReMon data
- Based on pandas and matplotlib
- Available at https://gitlab.in2p3.fr/bastien.gounon/plotemon

- Evolution of each metric over time, for a single job (ex: run_calexp)



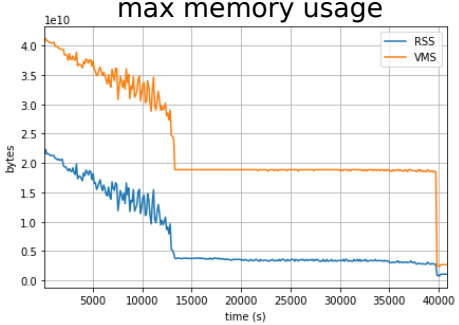




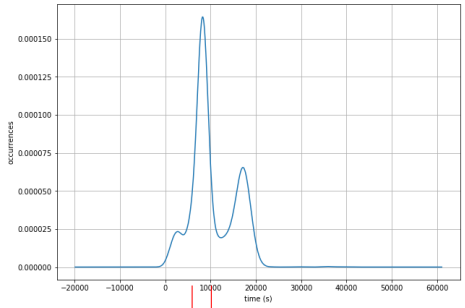
Visualization and analysis

- Compile metrics from many jobs to extract general tendencies

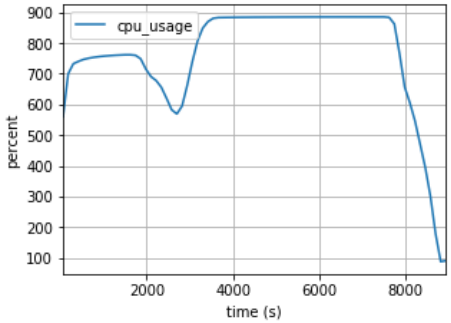
max memory usage

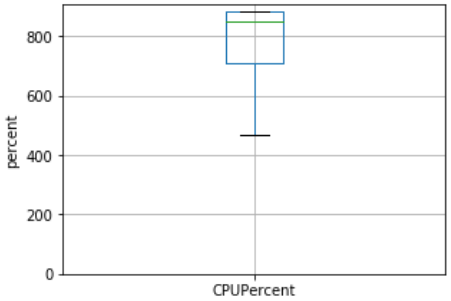


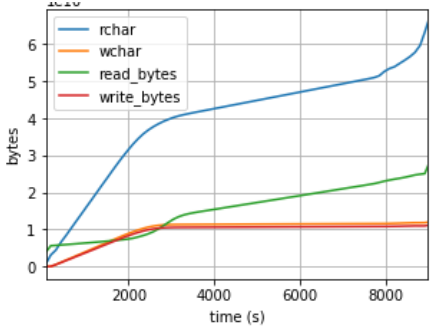
job duration repartition

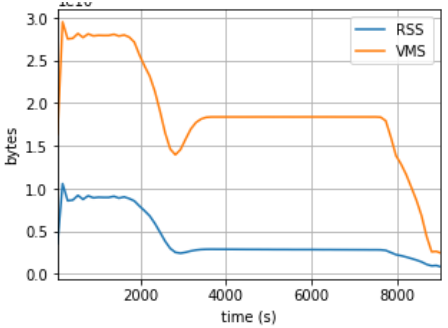


dataset of ~5000
run_calexp jobs









average behavior for
typical runs

Evolution

- JoReMon:
  - Collect network usage
  - Output data into a time-series database

- Visualization:
  - Dynamic web UI or Grafana dashboard instead of notebooks
  - More plots: we need your input !

Conclusion and questions

- JoReMon is a generic tool for resource usage collection on Linux
- Can easily be integrated in any workload
- Visualization tools are still in early version and are bound to improve
- If you are interested, get in touch

Thank you!