MAX-PLANCK-GESELLSCHAFT

MPCDF

MPCDF HPC Performance Monitoring System: Enabling Insight via Job-Specific Analysis

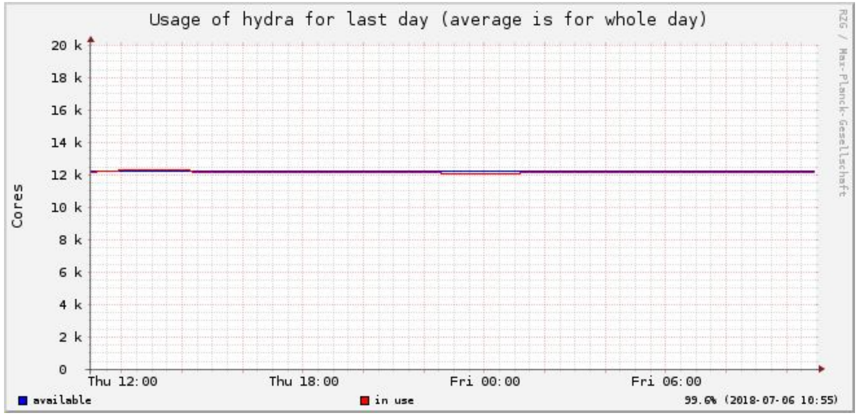
Luka Stanisic, Klaus Reuter

Max Planck Computing and Data Facility

PMACS workshop, Aug 26, 2019

Motivation

- HPC centers are well aware of statistics on HPC resource allocation, but what about the **actual utilization** of the resources?



- We know that there are many inefficient codes running on HPC systems, but we were lacking numbers to **quantify it**

Luka Stanisic, Klaus Reuter

Potential Beneficiaries

- **Computational scientists**
  - Use resources more efficiently to maximize resulting scientific knowledge
  - Optimize resource reservation batch scripts, compiler flags, load balancing, etc.
  - Ask application support for help
- **Application support**
  - Get a coarse-grain view of the jobs performance quickly, zoom with profilers later
  - Proactively approach users in need of assistance
- **System administrators**
  - Check the impact of software updates, security patches, hardware settings
  - Have historical performance data for a fair comparison
- **Management**
  - Analyze the actual utility of the investments in network, accelerators, fat-nodes, etc.
  - Steer decision for the procurements of future HPC system

Existing Solutions

- Monitoring solutions developed and used by other centers
  - LIKWID Monitoring Stack (Erlangen, Germany)
  - PerSyst (LRZ, Germany)
  - TACC Stats + XALT + REMORA (Austin, USA)
  - Ovis based on LDMS (SNL and NCSA, USA)
  - EDF-HPC (Paris, France)
  - etc.
- Pros: Documented, (some are) open source, proven to work
- Cons: Dependencies on libraries and kernel modules, customization to MPCDF, potential end-of-life issues

→ Motivation to develop our own simple and lightweight solution!

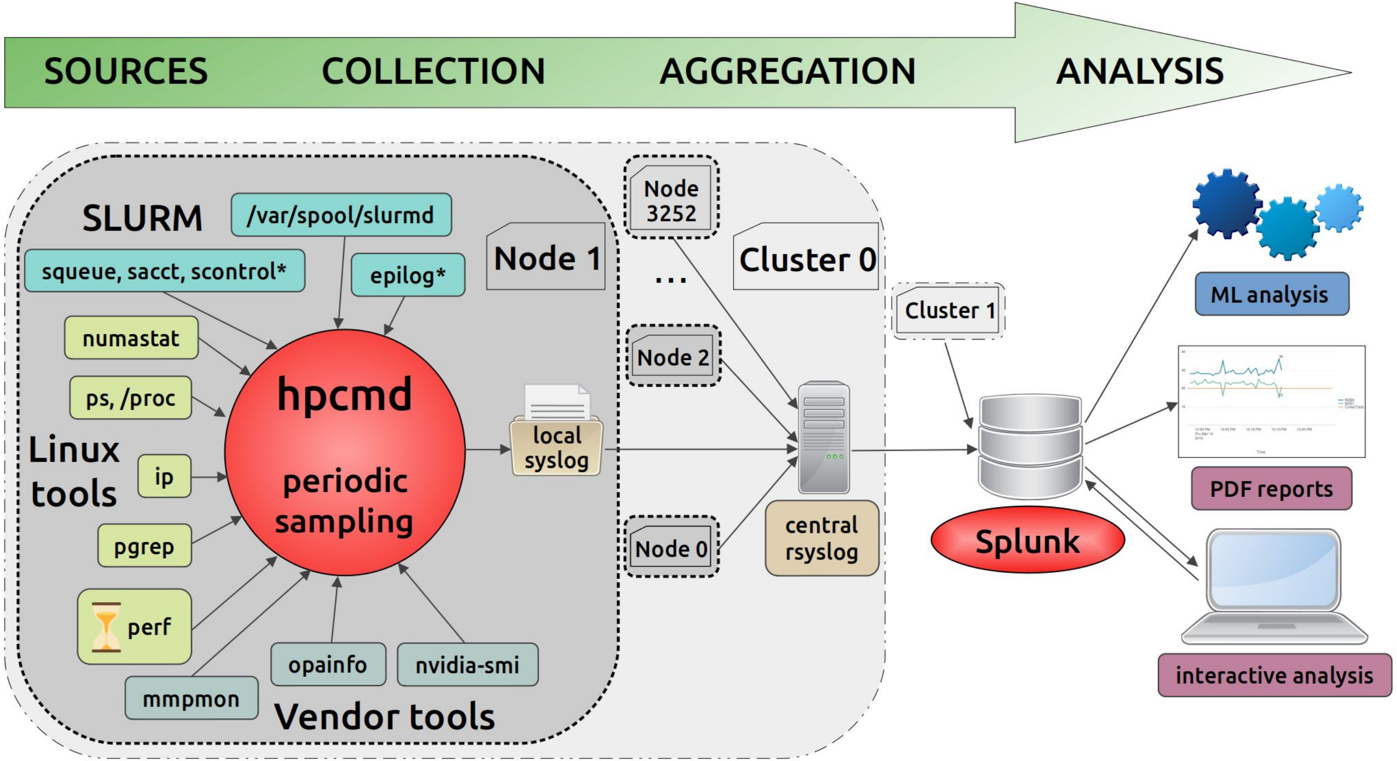
Necessary Components

1. Measure performance data
    a. Define appropriate metrics to quantify a job's performance
    b. Find appropriate measurement tools
2. Collect and parse the data from measurement tools
    a. Perform sample-based monitoring
    b. System-wide (default) or user-defined monitoring
    c. Overhead generated by monitoring must be negligible
3. Aggregate data from all compute nodes
    a. Communication layer
    b. Be careful with data volume and network load
4. Analyze, visualize and report performance data
    a. Web interface
    b. Ensure scalability up to large HPC systems
    c. Generate illuminating plots for application support and administrators
    d. Detect problematic jobs easily, finally apply "big data" analytics
    e. Make performance data available to the HPC users

Our Simple Approach

1. Measure performance data → standard Linux tools (top, perf, ps, etc.)
   a. Define appropriate metrics to quantify a job's performance
   b. Find appropriate measurement tools
2. Collect and parse the data from measurement tools → hpcmd: newly developed lightweight middleware
   a. Perform sample-based monitoring
   b. System-wide (default) or user-defined monitoring
   c. Overhead generated by monitoring must be negligible
3. Aggregate data from all compute nodes → rsyslog
   a. Communication layer
   b. Be careful with data volume and network load
4. Analyze, visualize and report performance data → Splunk and Python
   a. Web interface
   b. Ensure scalability up to large HPC systems
   c. Generate illuminating plots (for application support and administrators)
   d. Detect problematic jobs easily, finally apply "big data" analytics
   e. Make performance data available to the HPC users

Solution Architecture



1. Data Sources

- SLURM job-related information
  - Job id, user id
  - Executable name
  - MPI/OpenMP configuration
  - Environment
- Hardware counters, using perf (per socket)
  - Floating point instructions: Scalar, SSE, AVX, AVX512 → GFLOP/s
  - Memory bandwidth [GB/s]
  - Branch misses, instructions per cycle, etc.

  (Floating point instructions and memory bandwidth → roofline analysis)
- Memory using numastat (per socket) and ps (RSS)
- Network using opainfo or /proc
- GPFS filesystem using mmpmon
- Processes and threads, pinning, using ps
- GPU monitoring, using nvidia-smi

2. hpcmd - HPC Monitoring Daemon

- Newly developed python-based software daemon for collecting performance metrics
- Periodic sampling of each node and write-out to syslog
- Easy to configure and deploy, extend and maintain
- Extremely low overhead, no measurable impact on jobs
- Deployment via RPMs
- Modes of operation
    - Systemd service running permanently on each node, suspendable by the user
    - User-mode daemon running in the background of a user's job, fully re-configurable

3. Data Aggregation

- Syslog messages with performance data are written by hpcmd
- Messages are collected on an IO node via rsyslog; traffic goes via Ethernet, no overhead on the application which uses Infiniband or OmniPath
- Messages are finally transferred to Splunk, which uses a flat file-based storage

From our perspective: hpcmd writes a syslog message and data appears in Splunk, almost immediately!

```
<14>Aug 20 16:38:03 talos013 hpcmd_slurm_epilog.sh: HPCMD_CHECKPOINT="job_summary" jobid="36271" userid="sluka" groupid="rzg(4133)" partition="p.talos
jobstart="1566311803" jobend="1566311883" elapsed="80" maxrss="6" njobsteps="1" timelimit="1-00:00:00" min_empty_cores="31" exe="srun" AI="5.80778226
odes="1" GF="1.12201851852e-06" FP-SCALAR="53473" FP-VECTOR="0" cores="40" MEM_PEAK="40" SOCKET_BALANCE="0.529940731657" VEC_RATIO="0"
```

```
<14>Aug 20 16:36:49 talos013 hpcmd.exe: HPCMD_CHECKPOINT="job_start" jobid="36271" nodeid="0" userid="sluka" opmode="user" epoch="20" awake="18" jobna
start="1566311803.98" nnodes="1" omp_num_threads="" ntasks="1" omp_places="threads" hint="multithread" threadspercore="2" realmemory="190000.0" cores=
```

4. Analysis, Visualization & Reporting

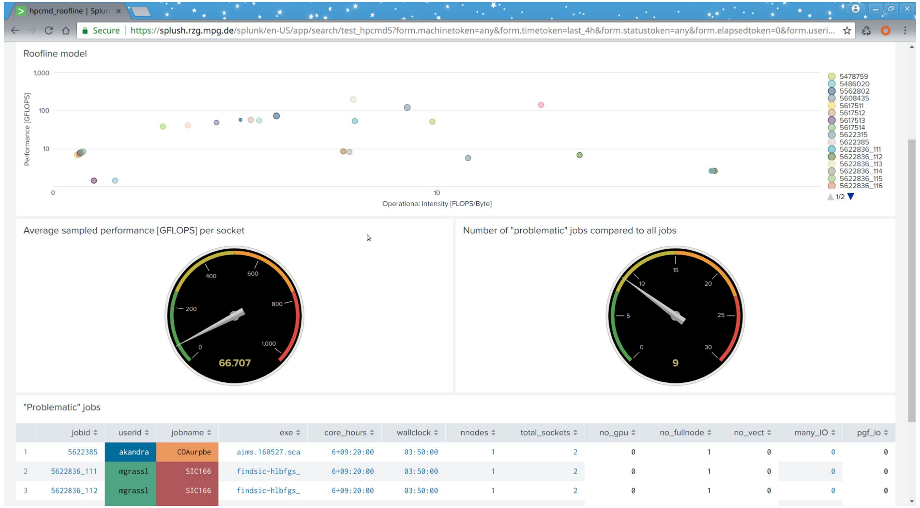
- Splunk: “Software for searching, monitoring, and analyzing machine-generated big data, via a Web-style interface. Splunk captures, indexes, and correlates real-time data in a searchable repository from which it can generate graphs, reports, alerts, dashboards, and visualizations.” [wiki]
- Private company, various types of licenses
- MPCDF was already using Splunk for other monitoring purposes
- Open source alternative: ELK stack, custom “handmade” frameworks
- Splunk is very powerful, intuitive to use, covers 99% of our needs!

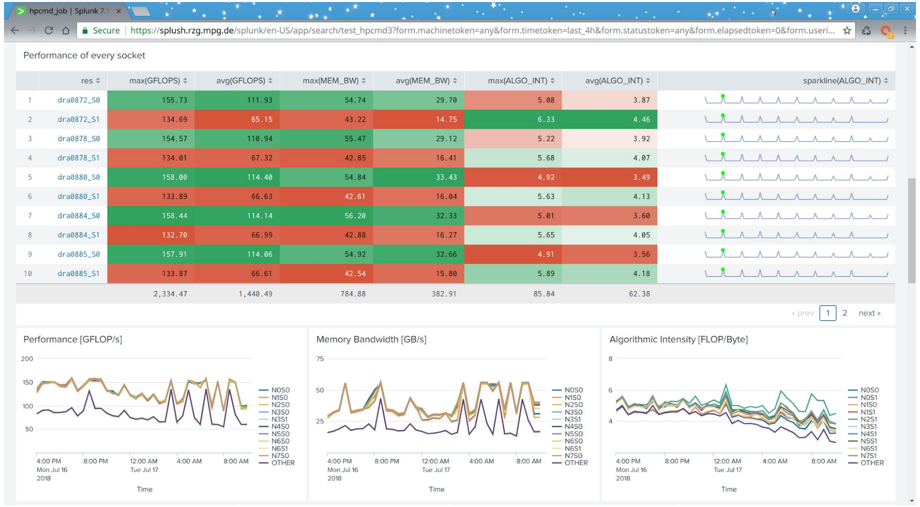
- Python: using mostly standard modules like pandas, numpy, matplotlib, scikit-learn, flask, etc.

Luka Stanisic, Klaus Reuter

Demos

- Roofline
- Job analysis
- Aggregated job analysis
- PDF reports
- Detecting ''black sheep'' jobs
- Top applications by core hours
- Helping with procurements
- Machine learning analysis





Summary

Status of HPC monitoring efforts at the MPCDF:

- MPCDF HPC Performance Monitoring System running on two large heterogeneous HPC systems
    - 4200 nodes (160.000 CPU cores), 3 generations of CPUs and GPUs, 2GB of data per day
- Performance report for each job accessible to the user via a unique PDF file
- Extensively used by HPC center employees, increasingly by the HPC system users

Lessons learned so far:

- Data already providing useful insight into performance issues otherwise hidden or unquantified
- Many jobs with non-optimal performance exist, often easy to detect and improve
- Coarse-grain monitoring data often enough to verify basic resource utilization assumptions

Ongoing Work

Technical aspects and improvements:

- Continue to develop and maintain the hpcmd middleware
- Investigate Nvidia DCGM tool for better GPU analysis
- Deploy our system to other, medium size, MPCDF clusters

How to additionally benefit from the gained knowledge best, internally and externally:

- Automatic performance verification of default application modules
- Apply big data analytics to performance data from large HPC clusters
    - Identify performance patterns and correlations, perform clustering, etc.
    - Completely automatize detection of jobs with non-optimal performance
- Use actual utilization data for future machine procurements

Thank you for your attention!

Questions?

Code: https://gitlab.mpcdf.mpg.de/mpcdf/hpcmd
Docs: http://mpcdf.pages.mpcdf.de/hpcmd

Luka Stanisic, Klaus Reuter