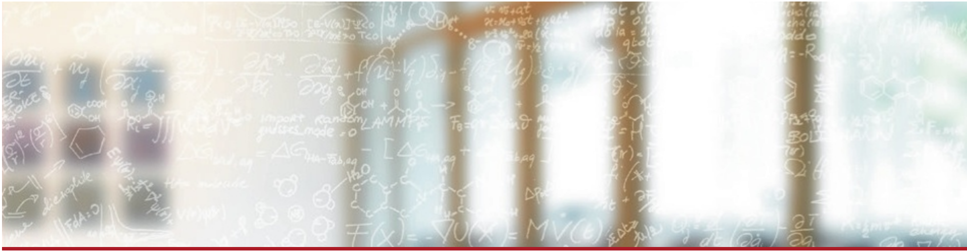
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Practical implementation of monitoring on Cray systems

CUG 2018, BoF 11C

V. H. Rusu, JG. Piccinali, G. Peretti-Pezzi

Tuesday, May 22nd 2018

Outline

- Our goals:
  - problems we are addressing / questions we want to answer
- Implementation goals and constraints:
  - Software: Open source, Specific stack ?
  - Hardware requirements, Cray specifically, HPC in general ?
- Implementation specifics (details, recipe to share)
  - Component / Data Flow Diagram of the system ()
  - Links to relevant information (recipes, papers, discoveries)
- Outcomes: (both positive and negative - learning experience)
  - Example(s) of scenario where the implementation was applied
  - Did it work? If not, why not?
  - What would you recommend doing differently?
- Q/A

Goals

- Share lessons learned in monitoring scientific application usage
  - Just application usage: what software our researchers use (or do not use) on our supercomputer ?
  - Only a subset of overall CSCS monitoring infrastructure
  - Only a subset of what the monitoring tool can do
  - Enough to expose (many) problems

Disclaimer

- We do scientific application support
  - reporting usage is our mission
  - we are neither slurm, nor monitoring tool developers
  - we are not data scientists

To err is human, but disaster requires a computer.

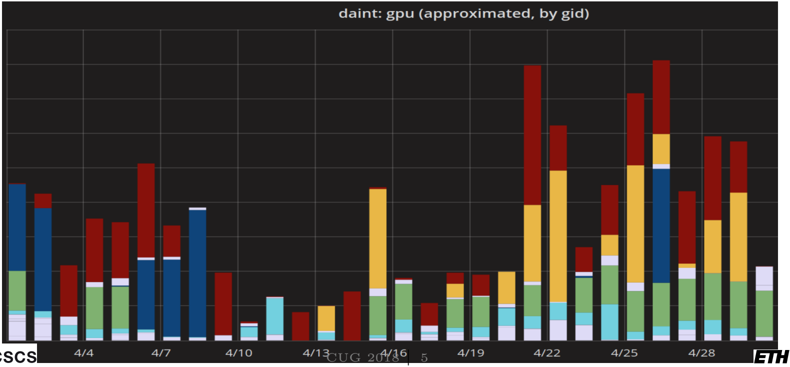
Piz Daint

https://www.cscs.ch/computers/piz-daint/

- Hybrid/Multicore Cray XC/40 and XC/50:
- Each XC/40 compute node hosts 2 Intel Broadwell CPUs
- Each XC/50 compute node hosts 1 Intel Haswell CPU and 1 NVIDIA P100 GPU
- Aries interconnect (dragonfly topology), Slurm

Typical usage reporting

- We track usage with:
  `http://github.com/Fahey-McLay/xalt.git`
  - **Xalt** intercepts the user link (ld) and job launcher (srun) user calls,
  - it maps program name & libraries to modulefile names,
  - it records complete list of environment vars, stores the results and provides reporting tools



CSCS specific implementation

- xalt: replaced blacklist of environment variables with whitelist (xalt_run_submission.py)
- xalt: added support for modulefile names longer than 64 characters (XALTdb.py)
- xalt: fixed gid capture plus additional slurm job fields (xalt_site_pkg.py)

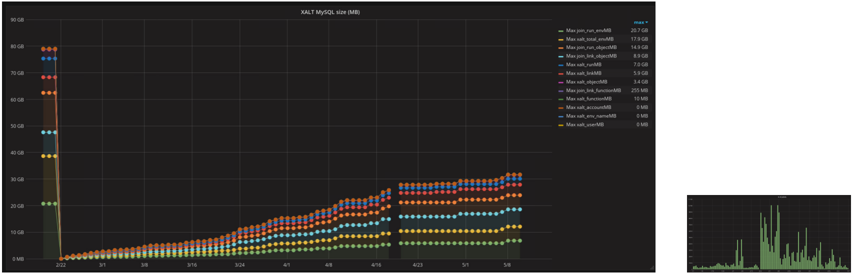
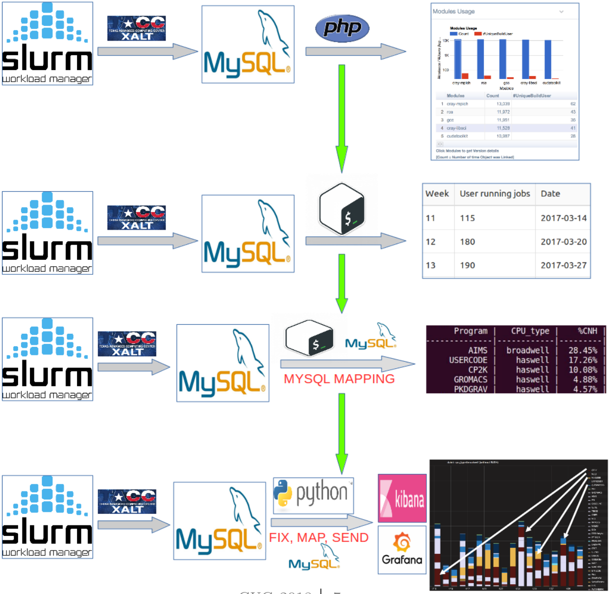


Figure: Left: Xalt's MySQL DB growth, right: number of jobs

Data flow



Modules Usage

	Modules	Count	#UniqueBuildUser
1	cray-mpich	13,039	62
2	rca	11,972	43
3	gcc	11,951	36
4	cray-libsci	11,528	41
5	cudatoolkit	10,987	28

Click Modules to get Version details

[Count = Number of time Object was Linked]

Week	User running jobs	Date
11	115	2017-03-14
12	180	2017-03-20
13	190	2017-03-27

```
     Program |   CPU_type |    %CNH |
-------------|------------|---------|
        AIMS |  broadwell |  28.45% |
    USERCODE |    haswell |  17.26% |
        CP2K |    haswell |  10.08% |
     GROMACS |    haswell |   4.88% |
     PKDGRAV |    haswell |   4.57% |
```

Current difficulties

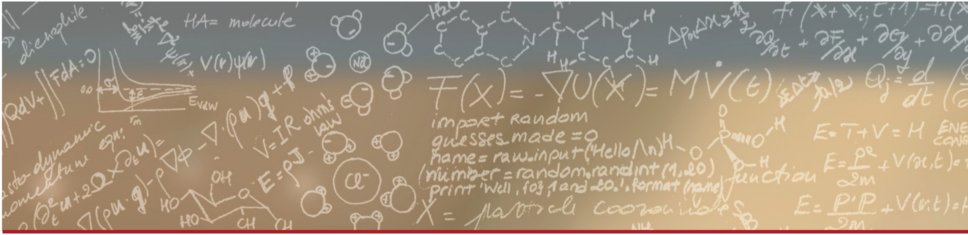
- Some slurm jobs are not captured
  - no call to `srun`
  - `srun --multi-prog`
  - `module unload xalt`
- Some slurm jobs may not be fully captured
  - Multi steps jobs - missing steps
  - Cancelled slurm jobs - wrong elapsed time (pap145)
- Difficult to map python based applications
- MySQL queries are slow - hundreds of jobs per day, and will get slower
- Duplicated data sets (SLURM and XALT databases)
- A lot of post processing needed to curate/validate XALT DB
- Application mapping has false positives and unmapped applications

What can we do differently?

- Instrument user code like XALT 2.0?
  - Solves the problem that some slurm jobs are not captured
  - Doesn't solve the cancelled slurm jobs - no end time
- Use native `srun` instead of XALT?
  - Decrease maintenance of additional DB
  - Duplicated data - data curation
- Deep learning for application mapping?
  - Use TensorFlow (ldd, objectdump, command line flags)?
  - Not enough data to train?
  - Biased data?
  - What about the unmapped user code? We will always need to talk to the users

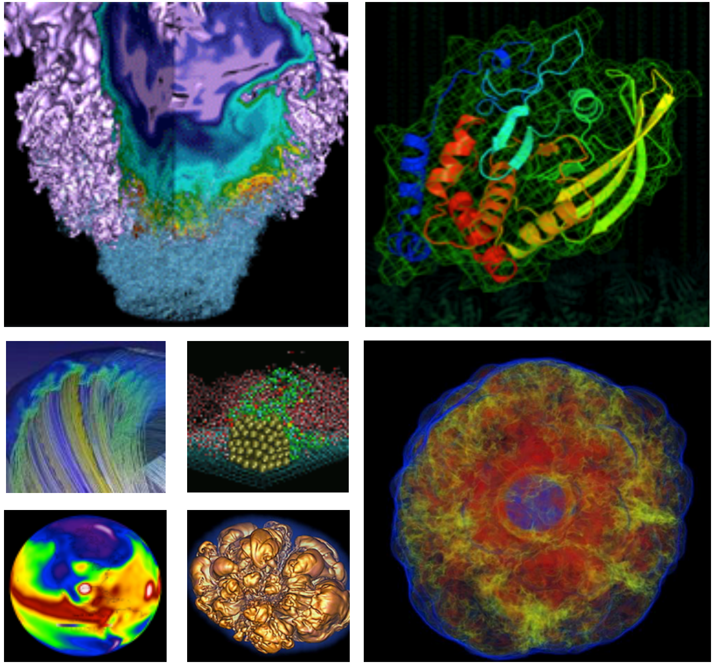
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Thank you for your attention

Data Visualization



NERSC

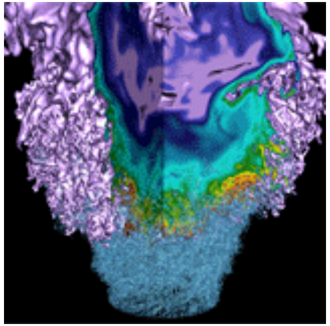
Cary Whitney

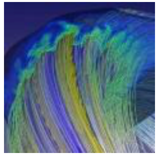
Stockholm/May 23, 2018

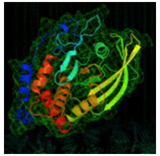
U.S. DEPARTMENT OF ENERGY | Office of Science

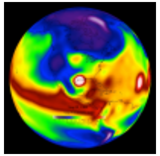
BERKELEY LAB

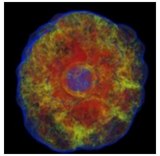
My Reference Definitions

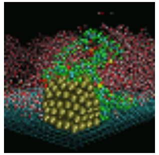












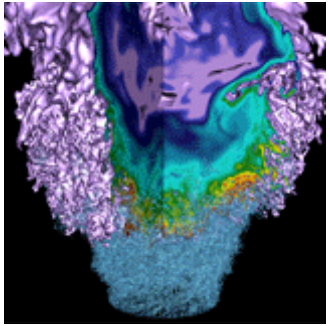
Definitions

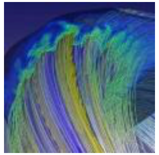
- Data - Event and time-series information about a system
- Data Collect - Structure and setup for the purpose of collecting data.
- Purpose for collected data
  - Monitoring - Ability to look at a data point
  - Knowledge/Understanding - Learning about what the data may be saying. New questions being asked.
  - Machine Learning - Grouping/correlation of data points
- Outcome of a data collected
  - Visualization - A method to display collected data
  - Alerts/notification/feedback - Outreach to a method/object to perform an action based on the data

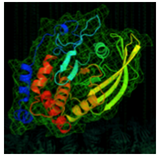
Data collect

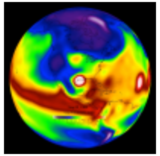
- Desire - Everyone wants one because they see the benefit
- Problems
  - High involvement of people
  - Hardware
  - Time
  - R&D on what is happening
  - Once there is an understanding, how to communicate it in a usable means
- But what part is important?
  - Collection of important data (Important is relative to your environment)
  - Action on collected data

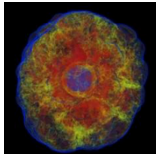
Collection of relative data

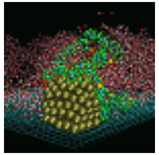








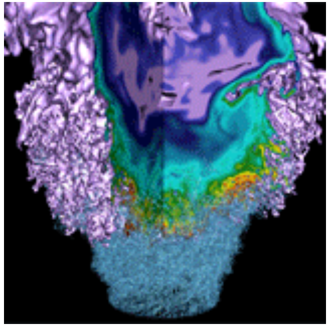


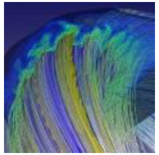


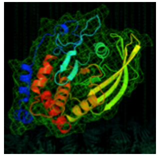
Collection

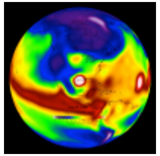
- Collection type, method, transport, storage, archive, etc of the data is defined by the site. References from other sites should be considered.
  - Vendor specific solutions may also work for some sites.
  - Size does not matter, just relevance to the site
- Large data collection
  - Allows R&D and discovery.
  - Concentrate on all data available
  - Wild idea: Collaborate with smaller organizations so they have the possibility to contribute. Their ideas are valuable also. :-)
- Small data collects
  - Contain only data for visualization and alerting/notification
  - Concentrate only on relevant data

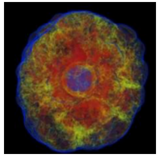
Outcome of data collected

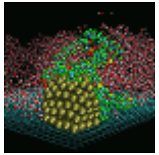












Relevant Data

- Need a good definition of what the data points are. Started
- Type of data - log/metric
  - Relevant collection rate for monitoring, for ML, for other applications
  - Log pattern
- What affects each data point?
- What the data point affects?
- Collection method for data. Partially started
  - Do sites need to run an application?
  - Is there a API to get access to the data?
  - Can there be a collection short form? (Small sites)
- GIT doc site?

Visualization

- Grafana - I hate to just name a product; the idea is a common visualization platform Started, can provide
  - Allows multiple data sources on a dashboard
  - Possible multiple data sources in a graph
  - Cray is using it for Lustre stats Started
  - IBM is using it for their GPFS stats Possible, other group
  - Does allow alerting on some data source - No Elastic at this time
  - Dashboards defined and exportable as JSON
  - Adapting a shared dashboard with others involve changing:
    - Data source
    - Maybe changing query in the graphs
  - Let’s create GIT repository and Best Practice on adapting dashboards

Alert/Notification/Feedback

- Grafana could be used by some sites. Elastic need something else.
- Nagios - Is this the default or a good standard?
  - Scripts can be shared
- Email/SMS of alerts - Are there better methods?
- Responses to issues
  - Are there best practises in dealing with known issues? Shared? GIT? Operation documents as a starting point.
  - Automated responses? Do we know enough yet? Goal?
    - Methods? Is this even more site specific? Can information be shared? A little testing
- Feedback into other applications? Do we know enough?
  - Share best practices? GIT? :-)

Issues (some)

- Data
  - From the understanding of different data points, sites can determine relevance
  - Can ML patterns be shared?
- Visualization
  - Getting permission from Cray/IBM to use their Grafana instance for other dashboards and data sources.
  - Our GPFS instance, IBM does not control the dashboards, Grafana is a site application.
  - Limited to what Grafana can display but open source
- Alerting/Notification
  - Data abstraction. Example: Allow same Nagios scripts to alert on a data point stored with different methods

Thank You

Reference Architecture for Monitoring HPC Systems

Goals of this effort

- Gain insight into system and application problems, resource constraints, usage trends, through monitoring
- Focus initially on a useful human view of data- ML or prediction can come later

Implementation goals and constraints

- Scalable, Performant, Minimize jitter, Reliable
- Multiple subsystems (server-side metrics/events, environmental, etc.)
- Portable to other systems, including non-Cray
- Software:
  - Widely adopted
  - Open source or free if possible
  - Leverage Containers
  - ELK or TICK? Graphana? Graylog
- Hardware requirements
  - Hot spool database for recent data that is off-system
  - Ideally, leverage system itself for long term data store and query, and possibly analysis
  - Ideally, seamless boundary between hot and cold spool for visualization interface

Infrastructure Model

- Data: Counter, *Event (requires log typing)*, and *Performance (requires polling)*
- Components:
    - Collect
    - Transport
    - Store
        - Archive, reduction, rotation, resampling
    - Query
        - Filter by source and timeframe (a job is one common example, but not limited to jobs, nor compute nodes within a job, or compute nodes themselves)
    - Visualize
        - Provide method to quickly navigate through data, drill down, cross reference, overlay, correlate, hover data, etc
    - Alerting

Implementation specifics

Still underway.

Outcomes

- Prototype is functional, useful
- Did it work?
  - Yes, but… it’s a prototype. Limited history, limited exposure level.
- What would you recommend doing differently?
  - Leverage containers where possible
  - Event storms
  - RDB table type (RDB at all?)
  - Refine data reduction/archival/rotation methods
  - Non-monolithic solution (components can be independently deployed and useful)
  - Separate mission critical function from non-critical (system doesn’t break if monitoring system is off)
  - Bin data by sample interval?

Q&A