Bright Cluster Manager
Scalable Accounting & Reporting for Compute Jobs

Michele Bertasi
Development Team Lead
21/05/2018
About Bright Cluster Manager

• Bright Cluster Manager:
  • Streamlines cluster deployments
  • Manages and healthchecks cluster after deployment
  • Integrates with OpenStack, Kubernetes, Spark, Ceph

• Cluster management daemon on every node (CMDaemon)

• Management interfaces:
  • GUI: Bright View
  • CLI: CMSH
  • API: JSON API (+ Python & C++ bindings)

• On-premise, off-premise, hybrid
• Easy to re-purpose nodes (also automatically based on workload)
• Rich collection of HPC/deep learning tools & libraries
About Bright & Cray

• Long history between Cray and Bright
  • Between ~2010 - ~2016: Bright used on cluster alongside XC systems for login nodes, storage nodes, data mover nodes
  • Since 2017: Bright standard on all CS systems

• In past completed 2 successful PoCs for Bright on XC

• Ambition still exists to expand from CS to XC series
Why Accounting?

- Systems are expensive
- Knowing how they are used helps
  - Capacity planning
  - Securing budget for future systems
  - Chargeback
  - Troubleshooting
What do we want to know?

• Who is using the resources?
• Who is using them poorly?
• How was the system behaving in a particular moment in the past?
• Are all the components of the system being used?
  • Are the jobs CPU bound or I/O bound?
  • Do we have enough network bandwidth?
  • What is the utilization on GPUs?
Two different approaches

• Monitoring
  • What is going on right now
  • Gives you visual insights

• Accounting
  • What has happened in a certain period
  • Gives you numbers that you can report to others

(e.g. power consumption for this user's jobs was 50 kW hour)
What do we monitor?

• A metric is a value that changes over time
• It helps you understand how different parts of your system behave
• Device based metrics
  • Network, CPU, load average, power consumption, ...
• Job based metrics
  • Cgroup metrics (CPU, memory, disk I/O)
  • Metrics and metadata from the Workload Management System
Monitoring evolution (I)

• Traditionally reporting was per-device (2005)
• Then we introduced per-job reporting (2015)
• Limitations
  • No way to aggregate data
  • No way to group and filter
Monitoring evolution (II)

• Introduced Workload manager Accounting & Reporting
• Features
  • Aggregate and filter metrics
  • Create graphs or reports for a particular period of time
  • Create reports over historic aggregation of data
  • Dynamic and flexible
Examples (I)

Memory usage by users

- Aggregate memory metrics by user
- Plot them over a period of time
Examples (II)

Current jobs' waiting time

<table>
<thead>
<tr>
<th>job_id</th>
<th>job_name</th>
<th>user</th>
<th>group</th>
<th>job_waiting_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>pi</td>
<td>bob</td>
<td>dev</td>
<td>69034 s</td>
</tr>
<tr>
<td>6</td>
<td>my_mpi_job</td>
<td>mike</td>
<td>ds</td>
<td>360 s</td>
</tr>
<tr>
<td>15</td>
<td>pi</td>
<td>bob</td>
<td>dev</td>
<td>10 s</td>
</tr>
</tbody>
</table>

- Single metric
- Take the last value
- Sorting
- Show them in a table
Examples (III)

CPU wall clock time used over the last week by account

<table>
<thead>
<tr>
<th>account</th>
<th>account_cpu_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectx</td>
<td>15300 CPU s</td>
</tr>
<tr>
<td>seismic</td>
<td>360 CPU s</td>
</tr>
<tr>
<td>drilling</td>
<td>369034 CPU s</td>
</tr>
</tbody>
</table>

- Aggregating over time
- Grouping by account
Examples (IV)

Power consumption of Bob's jobs over the last week

<table>
<thead>
<tr>
<th>power_usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>231 kWh</td>
</tr>
</tbody>
</table>

- Aggregation over time
- Filtering by a particular user
- Single number as a result
Grouping and filtering

• Metrics come with labels
  • They are arbitrary key value strings providing metadata
  • Specified by the metrics producer

• Examples of labels
  • User = bob
  • Job = job.15
  • Hostname = node123
  • Queue = high-priority

• Filtering and grouping can be done on labels
  • CPU usage only user="bob"
  • Average memory usage on nodes with category="bigmem"
Flexible reporting (I)

- Provide sensible defaults and ready to use dashboard
- Different people want different reports
  - Impossible to make everyone happy
  - Impossible to determine the needs of tomorrow
Flexible reporting (II)

- Standard interfaces and pluggable components
- PromQL as query language
  - Known in the industry, well documented
- Prometheus data sources
  - Allows integration of open-source metrics samplers
  - Easy to create new samplers
- Prometheus query APIs
  - Allows plotting from any compatible dashboard (e.g. Grafana)
PromQL (I)

- PromQL is a functional expression language
- Allows to select and aggregate time series data in real time
- Features
  - Labels selection
  - Arithmetic and comparison operators
  - Aggregation
  - Joins
  - Statistical functions
  - Sorting
  - ...
### PromQL (II)

#### Jobs' allocated nodes

**PromQL Query:**

```
job_metadata_num_nodes
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>GROUP</th>
<th>HOSTNAME</th>
<th>JOB</th>
<th>JOB ID</th>
<th>JOB_NAME</th>
<th>QUEUE</th>
<th>USER</th>
<th>WLM</th>
<th>TIME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>job_metadata_num_nodes</td>
<td>bob</td>
<td>mb-c-05-02... JobMet...</td>
<td>6</td>
<td>long.job</td>
<td>defq</td>
<td>bob</td>
<td>slurm</td>
<td></td>
<td>Tuesday, May</td>
<td>1</td>
</tr>
<tr>
<td>job_metadata_num_nodes</td>
<td>mike</td>
<td>mb-c-05-02... JobMet...</td>
<td>4</td>
<td>long.job</td>
<td>defq</td>
<td>mike</td>
<td>slurm</td>
<td></td>
<td>Tuesday, May</td>
<td>1</td>
</tr>
<tr>
<td>job_metadata_num_nodes</td>
<td>mike</td>
<td>mb-c-05-02... JobMet...</td>
<td>5</td>
<td>long.job</td>
<td>defq</td>
<td>mike</td>
<td>slurm</td>
<td></td>
<td>Tuesday, May</td>
<td>1</td>
</tr>
</tbody>
</table>
PromQL (III)

User's allocated nodes

```
sum by(user) (job_metadata_num_nodes)
```
PromQL (IV)

Associate power consumption with jobs

<table>
<thead>
<tr>
<th>hostname</th>
<th>pwr_consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>node001</td>
<td>250 W</td>
</tr>
<tr>
<td>node002</td>
<td>233 W</td>
</tr>
<tr>
<td>node003</td>
<td>245 W</td>
</tr>
</tbody>
</table>

job_id  hostname  user  is_running
5       node001    bob   0
6       node001    bob   1
6       node003    bob   1
7       node002    mike  1

user | user_power_watts |
-----|-----------------|
bob  | 495 W           |
mike | 233 W           |

job_id  hostname  user  job_power_watts
5       node001    bob   0
6       node001    bob   250 W
6       node003    bob   245 W
7       node002    mike  233 W
Architecture

Monitor → Store → Query

Compute node

Job 1
cgroup

Job N
cgroup

Bright Cluster Manager

store

PromQL engine

query

Monitoring DB

monitoring data
Usage

• Predefined queries
  • Sane defaults that work 90% of the cases

• Custom queries
  • Modify the defaults
  • Save new queries for later use
  • Queries on the fly to experiment

• Export the data to Excel, CSV
Accounting for end users (I)

- Jobs metrics are sensitive
  - Only the administrators can view everything
  - End-users by default are not allowed to see anything

- Planned for BCM 8.2
  - Allow end-users to see their metrics
  - Permissions define who can see what (e.g. only their own jobs, nothing, everything)
Accounting for end users (II)

• The user portal will add accounting & reporting
  • No need to be admin to see the monitoring data
• Allows users to get insights from their jobs
  • Job based metrics
  • Accounting and statistics
Scalability (I)

- Monitoring system currently scales to ~10k nodes in typical use.
- The load is a factor of:
  - # of nodes
  - # of jobs
  - # of metrics
  - Sampling interval
Scalability (II: storage)

• Metrics data is not stored indefinitely
  • Raw data limited to last period (time window, or # of samples)
  • Older data is consolidated (per day, week, month, …)

• Recording rules
  • Execute queries periodically
  • Aggregate data
  • Store data for longer periods

• Completely configurable
Availability

• Data is replicated across both head nodes
• At least one head node needs to be up
In the works

• Dedicated monitoring nodes
  • Increased scalability and reliability
  • Decrease load on head nodes
• Custom job metrics
  • Application specific data
• Parametrized queries
• Monitoring for end users
• More default metrics and queries
Conclusion

- Accounting & Reporting allows admins to gain insights
  - On how effectively the resources are used
  - By which groups of users
- Easy to start with defaults
  - No prior knowledge required
  - No additional software
- Flexible reporting
  - PromQL gives you infinite possibilities
- Standard interfaces
  - Plug your metrics
  - Plug your queries
  - Plug your dashboards
Bright offers a complete platform to get insights on your infrastructure...