Storage and use of metric data

Mike Showerman
Best Practice: Project Management

What is the Best Practice?

The Blue Waters Project Office manages the project via weekly meetings with project managers and all funded staff, regular meetings with vendors, ticket system, quarterly internal reviews, a risk register, external reviews, and external evaluations.

Why is it needed?

The goal is to ensure that the end users are provided with the best possible access to resources and services, and a supportive working environment for staff. The Blue Waters project involves funded staff, contracted vendors, support from other NCSA teams, and a distributed user base.

Who does it impact and when?

This activity is conducted on an ongoing basis with all of the Blue Waters staff, contracted vendors and partners, and NCSA staff, and end users.

Why is this a Best Practice?

It reduces knowledge isolation and helps ensure staff feel they are all part of the team. This results in greater staff productivity and user satisfaction by solving problems quickly with a team approach.
Infrastructure

- Compute/Service nodes
- Aggregator
- Aggregator
- Aggregator
- Aggregator
  - Full Raw Data Set
- Derived Data Set
  - Integrated System Console
- All System Data
  - HTTP
  - Lustre
Data descriptions

- File stores
  - About 90G of MSR data, 55G of node metric data per day
  - Data is raw format such as counters
- Integrated System Console Database
  - ~16G per day of node data per day, 4 days retained
  - Some data is preprocessed from counters to rates
    - Example: flop counters per core are converted to flop rate per node
    - Greatly improves query efficiency
  - All log and system data
- Access methods
  - SQL queries
  - CSV files in lustre (parallel tools to extract data)
  - Web interfaces (with image/raw data downloads)
  - Published the datasets
Current Database Data

- Gemini Link Statistics
  - All 6 directions
  - Link BW, %used, average packet size, %input queue stalls, %credit stalls, …
- Gemini/NIC Statistics
  - totaloutput_optA/B, total input, FMA output, bet output
  - SMSG
    - Number tx/rx rate, Bytes tx/rx rate
  - RDMA
    - Number tx/rx rate, Bytes tx/rx rate
  - IP over Gemini
    - Transmit/Receive rate
- Application library use
- MPI I/O operations from each application (Darshan)

- Node
  - Load average
    - Latest, 5min, running processes, total processes
- Flop rate
- Current free memory
- GPU
  - Utilization, memory used, temperature
  - Pstate, Power Limit, Power Usage
- Filesystems
  - For each home, projects, and scratch
    - Bytes/sec Read and write
    - Rate of Opens, closes, seeks
Long term analysis using file data and large parallel processing custom apps

Last 6 months of each compared:
Network Injection = 42% increase with TAS

2.57 TB/s
3.66 TB/s

Traditional  TAS Ramp Up  TAS
Easy Chart generator proof of concept
How do we use the metric data with the system data?... System
Contribution by job
Contribution by user
Contribution by project
The value of higher fidelity

Lustre write performance for 2 identical jobs
How is this all Done

- The initial ISC was just mysql
- Everything goes to the same database
- Metrics can be viewed by any property
  - Nodes in job
  - Nodes in apid
  - Node type
  - Nodes in jobs by jobname….
- Event data could be added to the charts or data from differing subsystems
- Flexible in query structure
IOPs Aggregates

LMTStats Scratch

- close
- connect
- create
- destroy
- disconnect
- getattr
- getxattr
- link
- ilog_init
- mkdir
- mknod
- notify
- open
- process_config
- quotactl
- reconnect
- rename
- rmdir
- setattr
- statfs
- unlink
First Use of Google Charts for Monitoring Job Data

Start: Nov 13, 2013 8:13 pm
End: Nov 16, 2013 7:44 pm
Overall Utilization Average: 82.83%
Node-Hours lost from Undersubmitted Workload: 21669.024 nodehours (1.34% avg)
Node-Hours lost by Down nodes: 823.854 nodehours (0.05% avg)
Node-Hours spent Draining: 255428.006 nodehours (15.76% avg)
Current Chart Resolution: 20 Minutes

1 person currently viewing this chart
Workload analysis
Collaboration with University at Buffalo
NSF funded project
  Analyzing usage by discipline, algorithm, library
  Analyzing usage patterns and changes over time
  Identify current and potential bottlenecks
Used data that spanned workload managers
(alps and torque), LDMS node based
statistics, XALT, Darshan, HPSS, sonnexion
Analysis required
  over 1.1 million core hours
  95 TB of input data from over 4.5M
  Produced 250 TB of data in 100M files
  Loaded into MySQL and MongoDB for
display with XDMoD

Link to 106 page report
-
Original plan for Realtime Detection Design
(for job progress and I/O)

- Datastore
- Query engine
  - Job status
  - Job metrics
- Decision Engine
- Exception Engine
- Job metric rollups and reports
- Job completion

- Notification System
- Status/display

Job and LDMS data