Lustre Monitoring with OpenTSDB

DataDirect Networks Japan, Inc.

Shuichi Ihara

2015/9/22

Lustre Monitoring Background

- **Lustre is a black box**
  - Users and Administrators want to know “what’s going on?”
  - Find “Crazy Jobs” in advance to prevent slow down.
- **Lustre statistics are valuable big data**
  - Not only monitoring and visualization, but also analysis
  - “Predictable” operations might become possible.
  - Helps optimize applications and data relocation.
- **Open Source based monitoring tool**
  - In general, open source is common in HPC system and it’s straightforward.
  - Various combinations are possible and will enable new use cases.

Components of Lustre Monitoring

- **Flexible data collector (monitoring agent)**
  - Collects statistics from Lustre /proc and sends them to the monitoring server over the network.
  - Runs on servers as well as client and routers.
- **Data store**
  - Receives stats from agents and stores them into the database.
  - This can be historic and query-able data
- **Interface for data analytics**
  - Collected data is not only for visualization, but also for analytics.
  - Application I/O analytics, filesystem analytics, etc.

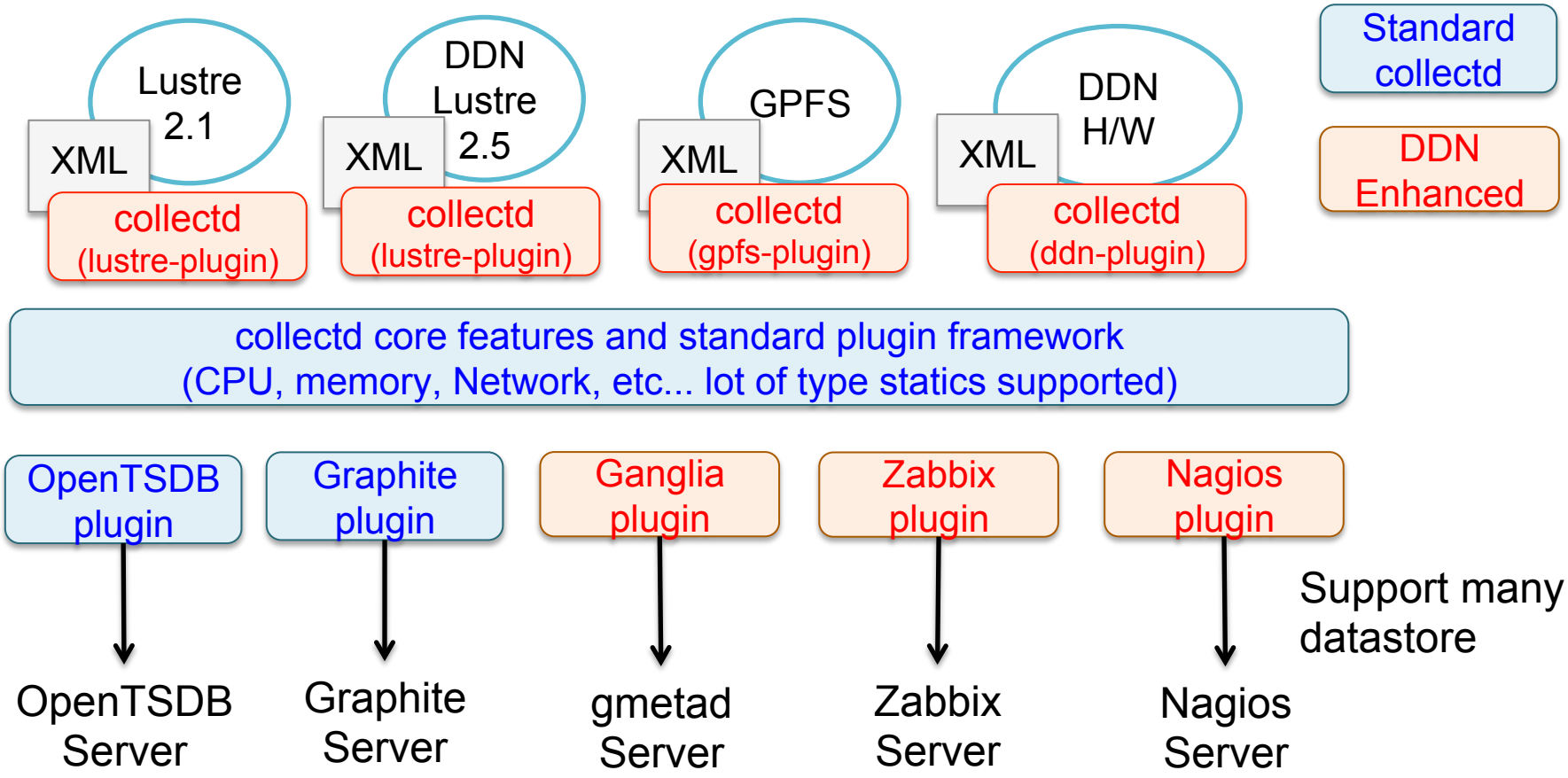
Flexible Data Collector (Monitoring Agent)

- **Lots of agents exist to collect Lustre performance statistics**
- **collectd is a reasonable options**
  - Actively developed, supported and documented
  - Running on many Enterprise/HPC system
  - Written in C with over 90 plugins are available.
  - Supports many backend database for data store.
  - Unfortunately, a Lustre plugin was not available, so we developed one.
  - Publish the lustre-plugin codes? Yes, we want to do so, but there were several discussions at LUG15. (e.g. stats in /proc or /sys in the future?)

Collectd Lustre Plugin Design

- **Our framework consists of two core components**
  - Core logic layer (Lustre plugin)
  - Statistics definition layer(XML file and XML parser)
- **XML for Lustre /proc information**
  - A single XML file for all definitions of Lustre data collection
  - No need to maintain massive error-prone scripts.
  - Extendable without core logic layer change.
  - Easy to support multiple Lustre version and Lustre distributions in the same cluster.
  - XML file automatically generated from m4 style definition file

Architecture of lustre-plugin and enhanced collected configuration



Scalable Backend Data Store

- **RDD and SQL based data stores do not scale**
  - RDD works well on small system, but writing 10M statics into files are very challenging (few million IOPS!)
  - SQL is faster than RDD, but still hits next level scalability, while database design is complex.
- **NoSQL based key-value stores are best**
  - OpenTSDB/Hbase.
  - Key, value and tags are easy adaption for Lustre statics data store. No need for complex database schema.
  - Retention awareness is needed – need policy for managing a stats data archive

OpenTSDB

- **Open Source "Time Series Database"**
  - OpenTSDB is a distributed DB (multiple TSDs), running on top of HBase.
  - It is scaling very well, depending on the underlying Hadoop cluster.
  - Trillion data point store is easily possible.
- **Many ingest and query options are available**
  - A lot of open source based agents are available (collectd has OpenTSDB plugin).
  - GUI (part of OpenTSDB) or other open source visualization tools (Grafana)
  - CLI, HTTP API

Adaption of Lustre stats into TimeSeries

- **Lustre and time series...**
  - Metrics
    - Defined each metric with each Lustre's proc entry (e.g. /proc/fs/lustre/mdt/*/md_stats)
  - Tags
    - Items in each proc entry (e.g. 'open', 'close', 'getattr' in md_stats)
    - Additional items
      - FS name, OST, MDT index number, etc..
  - Data Points
    - collectd sends packed metric, tags and value as a data point
    - (e.g.) tsdb_name md_stats, tsdb_tags optype=getattr fs_name=scratch3 mdt_index=MDT0000, value 2

1.7 Trillion Data Points in 24 hours

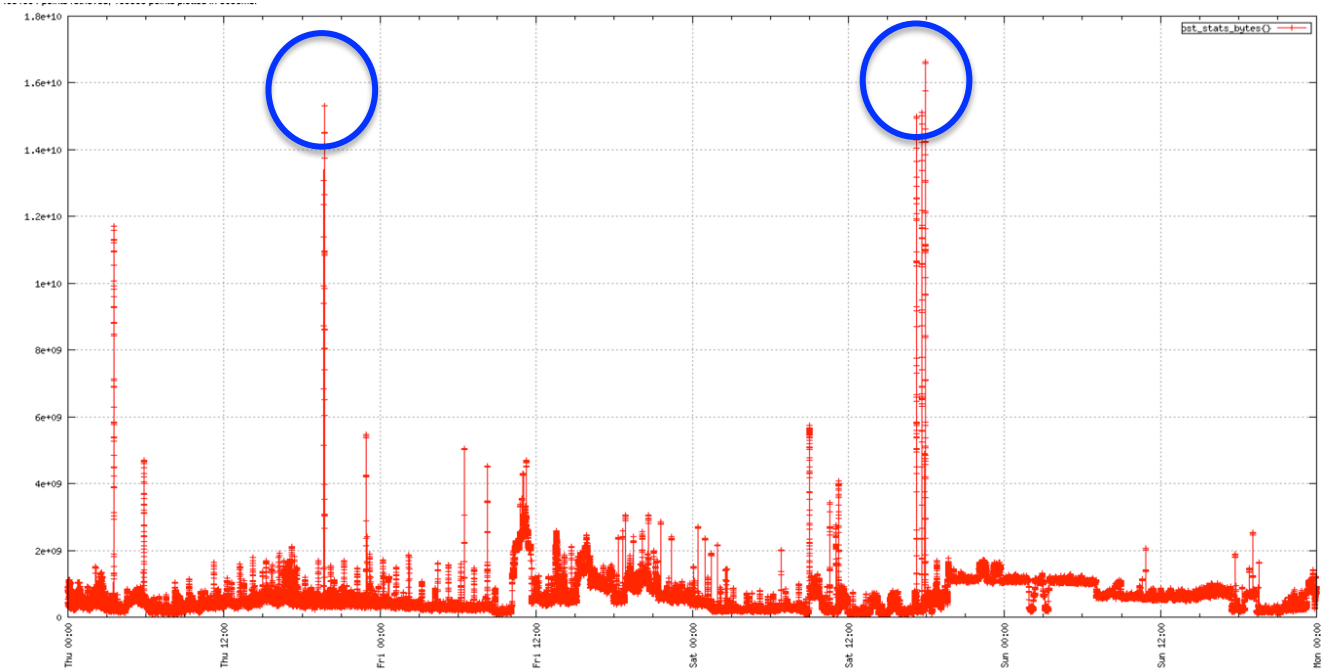
- **Developed “stress” plugin for collectd**
  - Generating “dummy” stats with collectd
  - For regression tests and benchmark tool of Lustre monitoring.
  - It works in conjunction with other collectd plugins.
- **Stress test on Lustre-plugin and OpenTSDB**
  - Setup two nodes hadoop/hbase cluster, with OpenTSDB runing on top of it.
  - 16 metrics “generators” (servers) generated total of 20M stats every second and send two the OpenTSDB servers.
  - Passed 24 hours stress test and stored 1.7 Trillion stats without any problems.

Application aware I/O monitoring

- **Scalable backend data store**
  - Now, we have scalable backend data store OpenTSDB.
  - Store any type of mercies whatever we want to collect.
- **Lustre Job stats is awesome, but need to be integration.**
  - Lustre JOB stats feature is useful, but administrator is not interested in I/O stats just only based on JOBID. (Array jobs. Job associates with another jobs, e.g. Genmic pipeline)
  - Lustre performance stats should be associated with all JOBID/GID/UID/NID or custom any IDs.

A realistic example on Lustre

- What happened here? Who, or What jobs caused burst I/O?
- But, often it's not a single job or a single user. Thus: What are top10 users/groups and jobs in an active Lustre I/O file system?



TopN Query with OpenTSDB

- **Implemented TopN Query based on OpenTSDB**

```
# topn -m mdt_jobstats_samples -r 1434973270 -s -t slurm_job_uid
Time(ms): 2015-06-22 20:41:25.000000        Interval: 5(s)
    rate |  fs_name | mdt_index | slurm_job_uid |  fqdn | slurm_job_id | slurm_job_gid |
15264.00 | scratch1 |   MDT0000 |          1044 | mds06 |        13290 |          1044 |
15076.80 | scratch1 |   MDT0000 |          1045 | mds06 |        13286 |          1045 |
13812.40 | scratch1 |   MDT0000 |          1049 | mds06 |            - |          1049 |
13456.80 | scratch1 |   MDT0000 |          1048 | mds06 |            - |          1048 |
 9180.80 | scratch1 |   MDT0000 |          1050 | mds06 |        13285 |          1050 |
 8909.40 | scratch1 |   MDT0000 |          1047 | mds06 |        13289 |          1047 |
 8779.60 | scratch1 |   MDT0000 |           502 | mds06 |            - |           503 |
 5049.00 | scratch1 |   MDT0000 |           501 | mds06 |        13291 |           502 |
```

- **Not only monitoring, but also a diagnostic tool!**
  - Quaries can be issued live of for specific time periods. (Rewind feature)
  - Lustre job stats associated with *all* UID/GID/JOBID/NID (or custom ID), rather than just one, is stored into OpenTSDB.

User reference(1)
Tokyo Institute of Technology

- **More than 1.2M stats into single monitoring server every 30 sec**
  - 14 Lustre servers, 154 OSTs for 3 lustre filesystems
  - 1700 clients mount all 3 filesystems
  - Lustre-2.1 is running. No jobstats! But collecting client based stats from “export” directory in /proc on Lusre servers.

    (Total stats = #OST * #Client * #metrics)
  - Demonstrated more than half Trillion stats stored into OpenTSDB over 6 months.

User reference (2)
Okinawa Institute of Science and Technology Graduate University

- **3PB Lustre filesystem**
  - Single Lustre filesystem (12 OSSs, 108 OSTs and over 400 clients)
  - Lustre-2.5 base
- **Lustre jobstats integrated with SLURM, running on production system**
  - Unique Lustre Jobstats configuration with Collectd Lustre plugin that runs on existing on Jobstats framework.
  - Collect jobs stats associated with all UID/GID/JOBID are stored into OpenTSDB.
  - TopN feature helps to find a root causes for unexpected burst I/O (i.e. who and what jobs caused problems).

Conclusions

- Developed new lustre plugin of collected. It's flexible, extendable and easy maintainable.
- Designed a Lustre monitoring framework based on a lustre collectd-plugin and OpenTSDB. The framework is running on several production systems to resolve today's lustre monitoring limitation.
- Demonstrated 1.7 Trillion data store into OpenTSDB in 24 hours. We will continue scalable testing for multi-Trillion data store in a few hours.
- Started the investigation of log data re-use for analysis.

Thank you!