Lustre文件系统的大规模性能监控与IO模式分析

顾政

gzheng@ddn.com

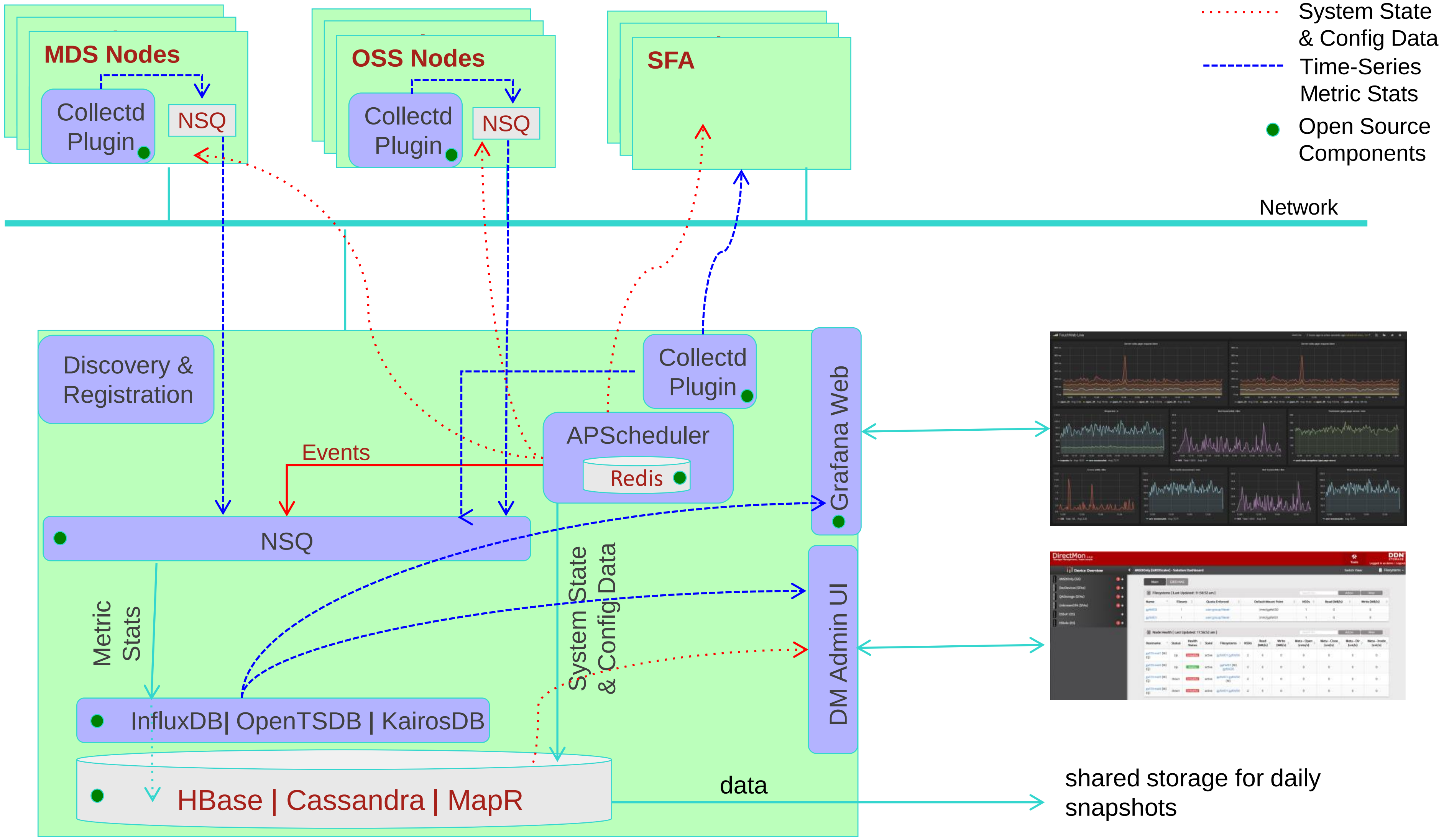
Background of Lustre Performance monitoring

- Activities on the Lustre are 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 could be possible.
  - It helps optimize applications and data relocation.
- Open Source based monitoring tool
  - In general, open source is common in the HPC system and it’s straightforward.
  - Various combination is possible and make new use cases.

C/S monitoring

- Collecting Data from target, usually it could be MDS/MGS, OSS, client.
- Sending collected data to persistent storage.
- Collected data could be reviewed by users friendly.(Time series, Rates etc.)

Standalone Configuration



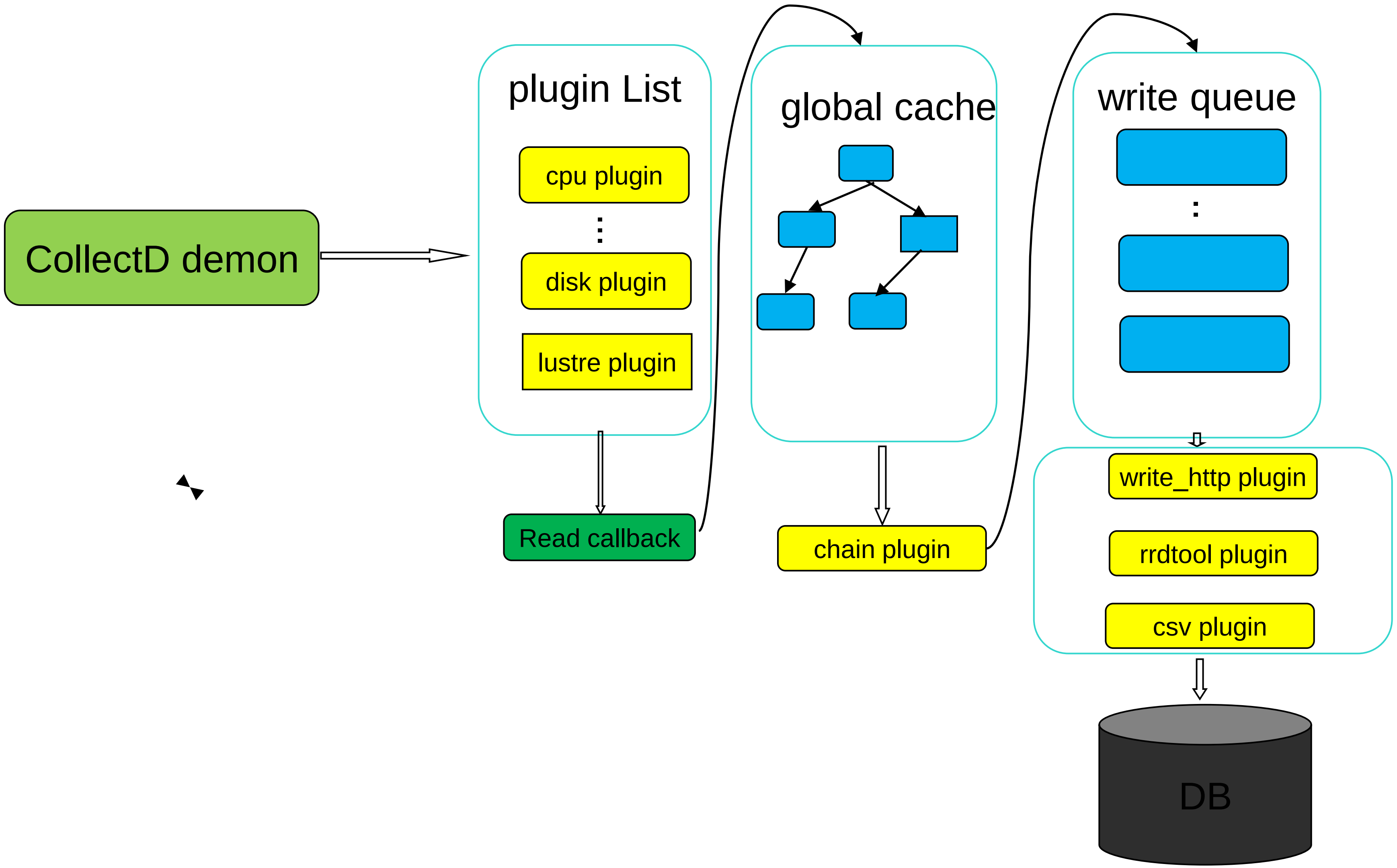
Components of Lustre Performance monitoring

- Data collecter
  - Collects statistics from Lustre /proc and sends them to monitoring server over the network.
  - Runs on servers as well as client and routers.
- Backend Storage
  - Receive stats from agents and store them into database.
  - It can be historical and query-able data
- Frontend
  - Collected data is not only visualized, but also analytics.
  - Application I/O analytics
- One-click setup
  - Native one-click script support, no need to assemble components manually, no worries about compatibility

Flexible data collector

- A lot of agents existed to collect Lustre performance statistics
- Collectd is one of reasonable options
  - Actively developed, supported and documented
  - Running at many Enterprise/HPC system
  - Written in C and over 100 plugins are available.
  - Supports many backend database for data store.
  - Unfortunately, plugin for lustre is not available, but we made it!

A glance at Collectd



Scalable backend data store

- RDD and SQL based data store dose not scale
  - RDD works well on small system, writing 10M statics into files are very challenging (few million IOPS!)
  - SQL is faster than RDD, but still hit next level scalability. And it’s complex to make database deign.
- NoSQL based key-value store shines
  - InfluxDB/Hbase. KairosDB/Cassandra
  - key, value and tags are easy adaption for Lustre statics data store. No need complex database schema.
  - Need to be aware of managing for statics data archiving. (retention)

Frontend – Why Grafana ?

- Visualize

  Heatmaps, histograms, graphs to geomaps..

- Alert

  Seamlessly define alerts where it makes sense

- Unify

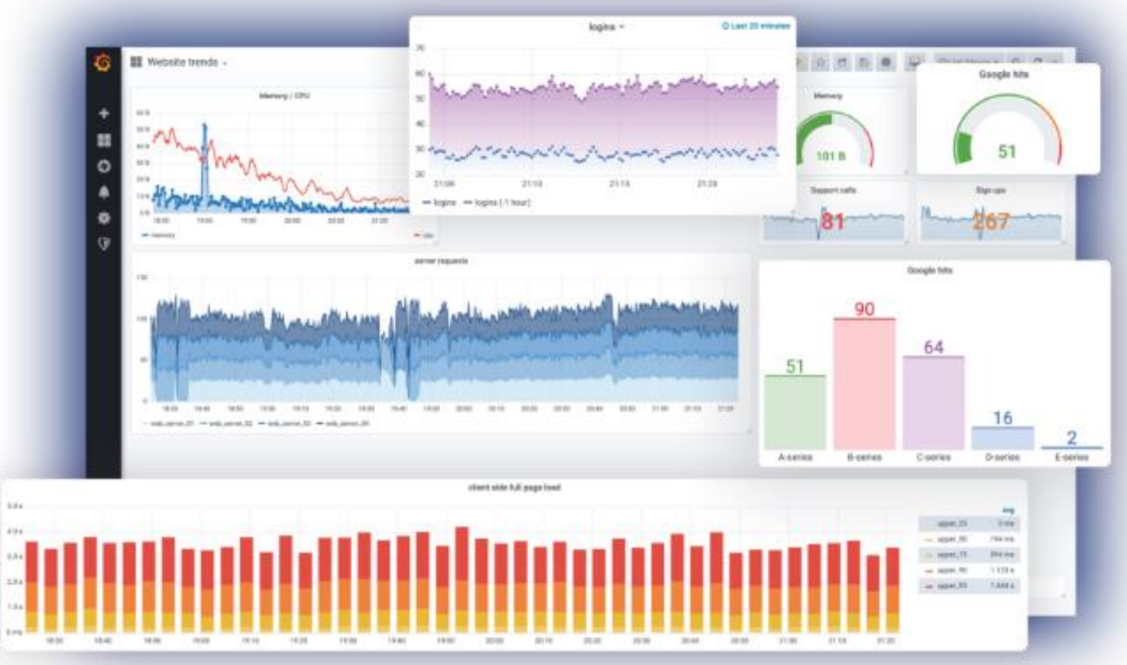
  Supports InfluxDB, Graphite, Elasticsearch, OpenTSDB and Prometheus.

- Extend

  Easy to customize dashboards and plugins

- Open

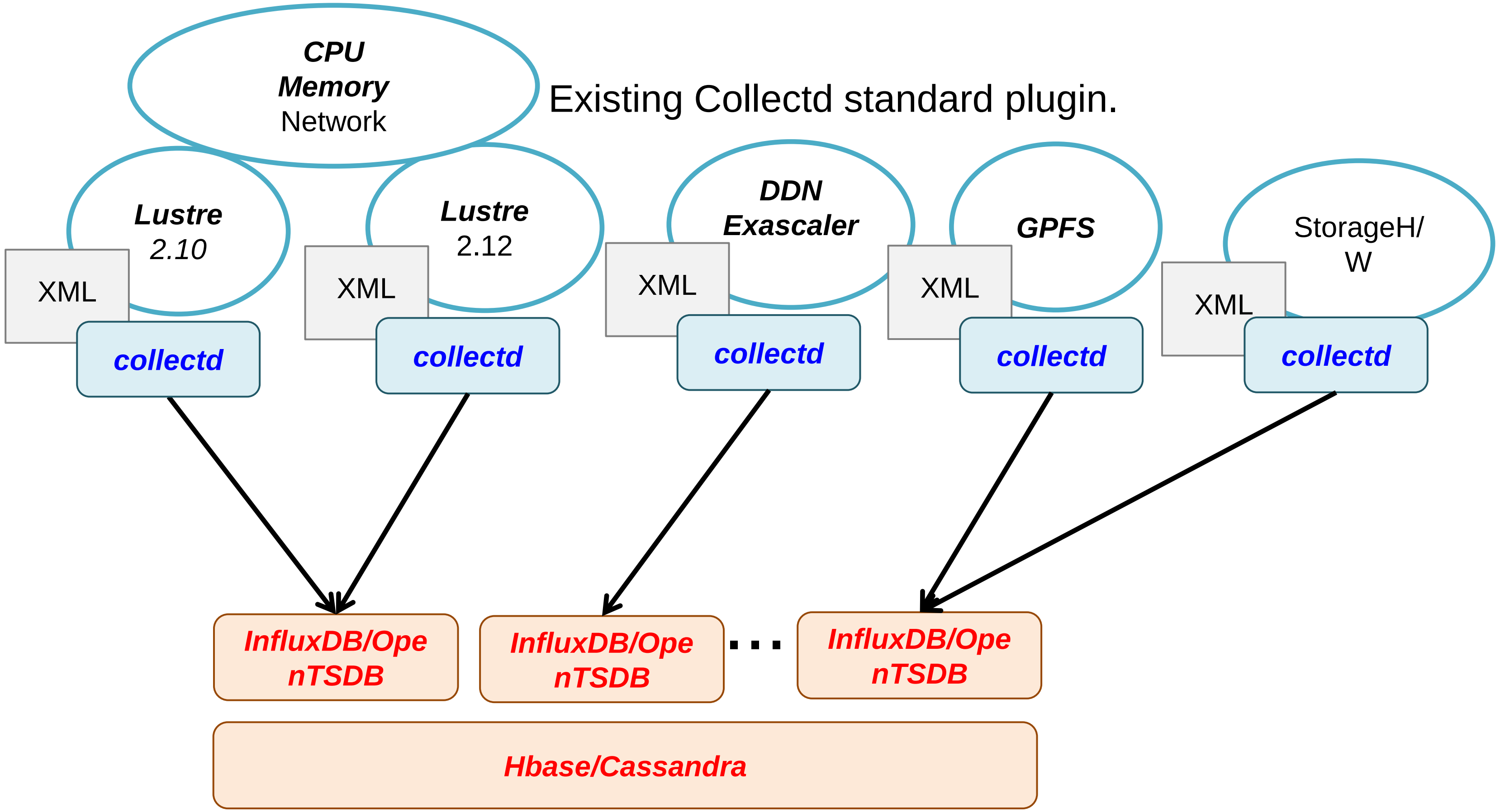
  Completely open source, and backed by a vibrant community



Deign of plugin for lustre in collectd

- A framework consists of two core components
  - Common platform, filedata plugin, collect data by reading and parsing a set of files (not only Lustre)
  - Statistics definition layer(XML file and XML parser)
- Defined 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.

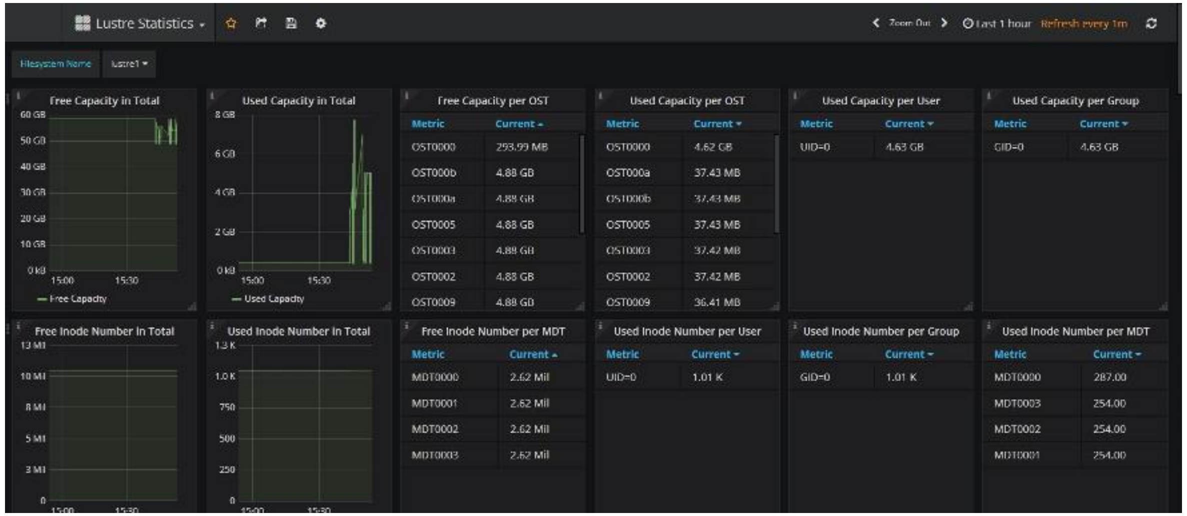
Architecture of lustre-plugin



Application aware I/O monitoring

- Scalable backend data store
  - Now, we have scalable backend data store InfluxDB/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.

Pictures of Lustre PerfMonitoring at customer site



Free Capacity per OST

Metric	Current ▲
OST0000	293.99 MB
OST000b	4.88 GB
OST000a	4.88 GB
OST0005	4.88 GB
OST0003	4.88 GB
OST0002	4.88 GB
OST0009	4.88 GB

Used Capacity per OST

Metric	Current ▼
OST0000	4.62 GB
OST000a	37.43 MB
OST000b	37.43 MB
OST0005	37.43 MB
OST0003	37.42 MB
OST0002	37.42 MB
OST0009	36.41 MB

Used Capacity per User

Metric	Current ▼
UID=0	4.63 GB

Used Capacity per Group

Metric	Current ▼
GID=0	4.63 GB

Free Inode Number per MDT

Metric	Current ▲
MDT0000	2.62 Mil
MDT0001	2.62 Mil
MDT0002	2.62 Mil
MDT0003	2.62 Mil

Used Inode Number per User

Metric	Current ▼
UID=0	1.01 K

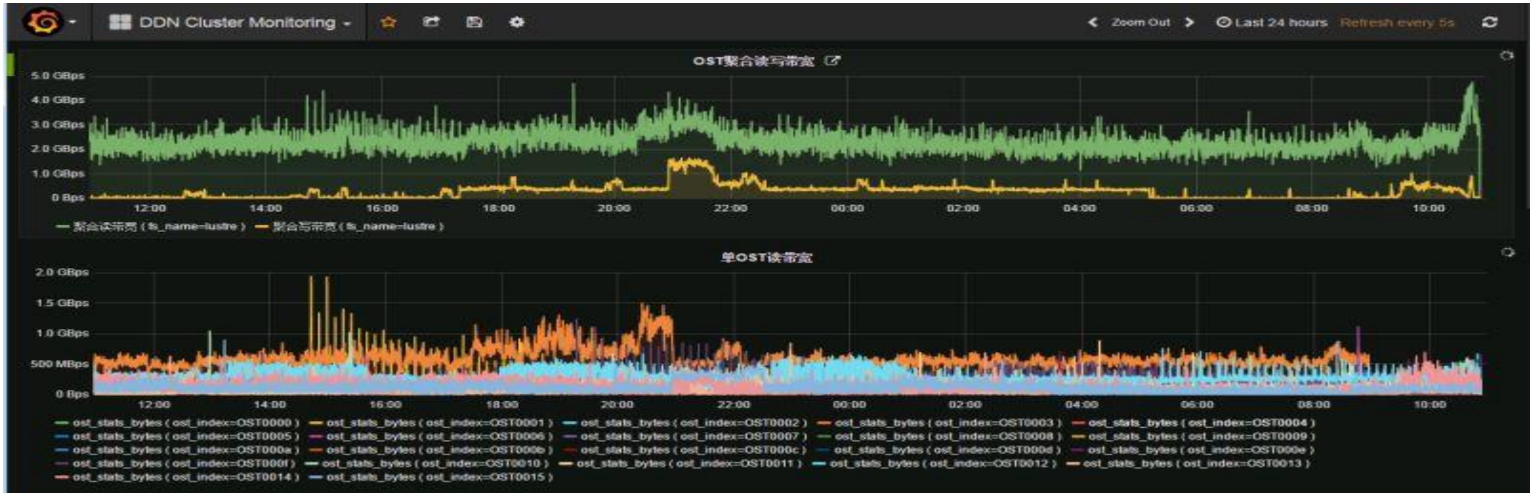
Used Inode Number per Group

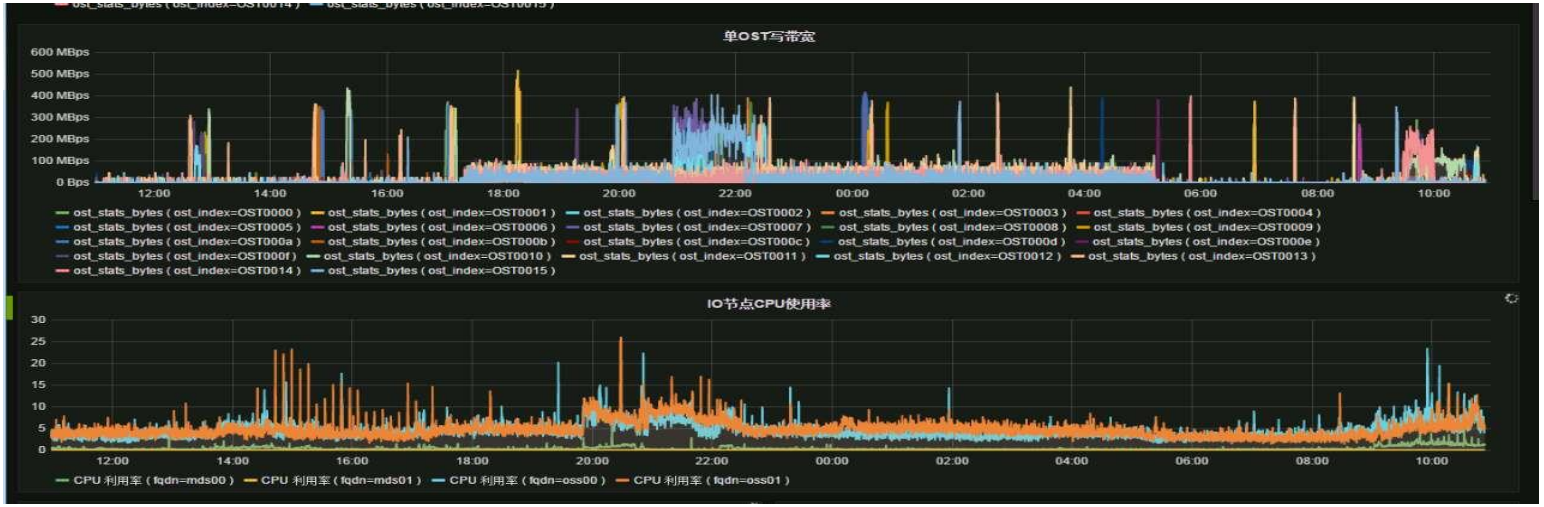
Metric	Current ▼
GID=0	1.01 K

Used Inode Number per MDT

Metric	Current ▼
MDT0000	287.00
MDT0003	254.00
MDT0002	254.00
MDT0001	254.00

Pictures of Lustre PerfMonitoring at customer site


DDN Cluster Monitoring
Zoom Out
Last 24 hours
Refresh every 5s
OST聚合读写带宽
5.0 GBps
4.0 GBps
3.0 GBps
2.0 GBps
1.0 GBps
0 Bps
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
08:00
10:00
聚合读带宽 (fs_name=lustre)
聚合写带宽 (fs_name=lustre)
单OST读带宽
2.0 GBps
1.5 GBps
1.0 GBps
500 MBps
0 Bps
ost_stats_bytes (ost_index=OST0000)
ost_stats_bytes (ost_index=OST0001)
ost_stats_bytes (ost_index=OST0002)
ost_stats_bytes (ost_index=OST0003)
ost_stats_bytes (ost_index=OST0004)
ost_stats_bytes (ost_index=OST0005)
ost_stats_bytes (ost_index=OST0006)
ost_stats_bytes (ost_index=OST0007)
ost_stats_bytes (ost_index=OST0008)
ost_stats_bytes (ost_index=OST0009)
ost_stats_bytes (ost_index=OST000a)
ost_stats_bytes (ost_index=OST000b)
ost_stats_bytes (ost_index=OST000c)
ost_stats_bytes (ost_index=OST000d)
ost_stats_bytes (ost_index=OST000e)
ost_stats_bytes (ost_index=OST000f)
ost_stats_bytes (ost_index=OST0010)
ost_stats_bytes (ost_index=OST0011)
ost_stats_bytes (ost_index=OST0012)
ost_stats_bytes (ost_index=OST0013)
ost_stats_bytes (ost_index=OST0014)
ost_stats_bytes (ost_index=OST0015)


单OST写带宽
600 MBps
500 MBps
400 MBps
300 MBps
200 MBps
100 MBps
0 Bps
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
08:00
10:00
ost_stats_bytes (ost_index=OST0000)
ost_stats_bytes (ost_index=OST0001)
ost_stats_bytes (ost_index=OST0002)
ost_stats_bytes (ost_index=OST0003)
ost_stats_bytes (ost_index=OST0004)
ost_stats_bytes (ost_index=OST0005)
ost_stats_bytes (ost_index=OST0006)
ost_stats_bytes (ost_index=OST0007)
ost_stats_bytes (ost_index=OST0008)
ost_stats_bytes (ost_index=OST0009)
ost_stats_bytes (ost_index=OST000a)
ost_stats_bytes (ost_index=OST000b)
ost_stats_bytes (ost_index=OST000c)
ost_stats_bytes (ost_index=OST000d)
ost_stats_bytes (ost_index=OST000e)
ost_stats_bytes (ost_index=OST000f)
ost_stats_bytes (ost_index=OST0010)
ost_stats_bytes (ost_index=OST0011)
ost_stats_bytes (ost_index=OST0012)
ost_stats_bytes (ost_index=OST0013)
ost_stats_bytes (ost_index=OST0014)
ost_stats_bytes (ost_index=OST0015)
IO节点CPU使用率
30
25
20
15
10
5
0
CPU 利用率 (fqdn=mds00)
CPU 利用率 (fqdn=mds01)
CPU 利用率 (fqdn=oss00)
CPU 利用率 (fqdn=oss01)

One story about Lustre PerfMonitoring at OIST

Lustre cluster configuration :

- 3PB Lustre filesystem (12 x OSS, 400 x client)
- Lustre jobstats integrated with SLRUM and running on the production system

Lustre PerfMonitoring configuration:

- Unique Lustre Job stats configuration with Collectd Lustre plugin and runs on existing on Jobstats framework.
- Collect jobs stats associated with all UID/GID/JOBID and store them into OpenTSDB.

With the help of Lustre PerfMonitoring, customer found out the root cause so quickly why unexpected burst I/O happened , which they suffered from for a long time.

Whamcloud

Thank You!

DDN®
STORAGE