Visualization of dCache accounting information with state-of-the-art Data Analysis Tools

Tigran Mkrtchyan for DESY dCache operating Team
The Flow (typical)

Collector → Parser → 

```
cat  |  awk  |  grep  |  gnuplot
```
Result

Bytes Read (365 days since Thu Mar 06 00:00:00 CST 2014)

Time
Scaling problems

> ~20GB billing files/day
> ~50.000.000 records/day
  ▪ ~500 records/sec
> 7 dCache instances
> need to adopt scripts for different needs
> need for a 'State at Glance'
The Flow

Collector

Parser

Processor

Selector

Visualizer

logstash

elasticsearch

kibana
Logstash

> Collect logs from any source
> parse them
> gets the right timestamp
> index them
> and move it into a central place
Logstash anatomy

```ruby
input {
  # read log events
}

filter {
  # parse, fix formats, mutate
}

output {
  # store processed events
}
```
Logstash, single liner

```
$ echo "hello logstash" | logstash -e 'input { stdin{} } output { stdout {codec => rubydebug} }'
{
    "message" => "hello logstash",
    "@version" => "1",
    "@timestamp" => "2016-03-06T22:49:37.797Z",
    "host" => "dcache-lab"
}
```
Real life example

}
filter {
    grok {
        match => [ "message", "%{TRANSFER_CLASSIC}" ]
        remove_field => [ "message" ]
    }
}

date {
    match => [ "billing_time", "MM.dd HH:mm:ss" ]
    timezone => "CET"
    remove_field => [ "billing_time" ]
}
}
> Regexp like syntax
> Lot of ready patterns for common cases
> supports labels and types
[00009A23BB6D280F46A7A6C12AC67F5EA897,59419220]
[0038000000000000000559888,46305280]

PNFSID_NEW (?:[A-F0-9]{36})
PNFSID_OLD (?:[A-F0-9]{24})
PNFSID %{PNFSID_OLD}|%{PNFSID_NEW}

PNFSID_SIZE \[%{PNFSID:\texttt{pnfsid}},%{\texttt{NONNEGINT}:size:int}\]
{DCap-3.0,131.169.74.175:34232}

PROTO (?:%{DATA}-[0-9]\.[0-9])

PROTOCOL \{%{PROTO:proto}(:)(%{IPORHOST:remote_host})(:)(%{NONNEGINT:remote_port:int})}
TRANSFER_CLASSIC %{BILLING_TIME:billing_time} %
{CELL_AND_TYPE} %{PNFSID_SIZE} %{PATH} %{SUNIT} 
{TRANSFER_SIZE} %{TRANSFER_TIME} %{IS_WRITE} 
{PROTOCOL} %{DOOR} %{ERROR}
Real Life example

{
    "@version" => "1",
    "@timestamp" => "2016-03-02T06:35:49.000Z",
    "type" => "dcache-billing",
    "host" => "ani",
    "path" => "/var/lib/dcache/billing/2016/03/billing-2016-03-02.log",
    "pool_name" => "dcache-desy23-05",
    "bill_type" => "transfer",
    "pnfsid" => "00009A23BB6D280F46A7A6C12AC67F5EA897",
    "size" => 59419220,
    "file_path" => "/pnfs/desy.de/desy/dcache.org/2.1/dcache-server_2.1.1-1_all.deb",
    "sunit" => "desy:generated@osm",
    "transfer_size" => 90112,
    "transfer_time" => 1195,
    "is_write" => "false",
    "proto" => "Http-1.1",
    "remote_host" => "dcache-infra03.desy.de",
    "remote_port" => 0,
    "payload" => ".:WebDAV-dcache-door-desy13:webdav-dcache-door-desy13Domain:",
    "initiator_type" => "door",
    "initiator" => "WebDAV-dcache-door-desy13@webdav-dcache-door-desy13Domain:1399012548236-1399012548243",
    "error_code" => 0
}

And store it in....

```json
output {
    elasticsearch {
        host => "elastic-search-master-node"
        index => "logstash-%{+YYYY.MM.dd}"
    }
}
```
Elasticsearch

> Open-source full-text search engine
> Schema-free JSON documents
> Powerful JSON based REST-API
> Distributed
  ▪ data can be divided into shards
  ▪ each shard can have zero or more replicas
> Node can be Master-node, Data-node or both
> Can be used as a NoSQL database
Document, Index and type

> Document is a basic unit of information
> Documents are expressed in JSON
> Each log entry corresponds to a document
> Index is a collection of documents
> An index is identified by a name (or alias)
> Name is used to refer to the index when performing actions
> Type is a logical category/partition of an index
> Type is defined for documents that have a set of common fields

(something like DATABASE (index), ROW(document) and TABLE(type) in RDBMS)
Shards and Replicas

> Index can be subdivide into multiple pieces

> Each piece called shard

> Each shard is an independent "index" and can be hosted on any node in the cluster.
  - allows horizontally split/scale data volume
  - allows distribute operations across shards

> You can make one or more copies of index’s shards called replicas
  - provides high availability in case a shard/node fails
  - allows to scale out search volume/throughput since searches can be executed on all replicas in parallel
CURD

> REST API

- **POST** – create document, index
- **GET** – search/read document
- **PUT/PATCH** – update document
- **DELETE** – delete document, index
Kibana

> Flexible analysis and visualization platform
> Real-time summary and charting of streaming data
> Intuitive interface for a variety of users
> Instant sharing and embedding of dashboards
Get started

> Dump data into elasticsearch
> Use discovery panel (or simple dashboard in Kibana3)
> Play with data
  ▪ search and aggregate
The building blocks

> Search
> Aggregation
> Visualization
Example
Example

![Graph showing visualization of dCache accounting information](image-url)
Example

Legend
- error_code: 0
- error_code: > 0

Count

@timestamp per 30 seconds
Dashboard

> A collection of visualizations

> Visualizations may use different 'data sources'

> A search in a dashboard affects all visualizations
Search in dashboard

![Kibana dashboard with search query: bill_type.transfer AND host.raw.dcache-se-atlas*]
Transfers at glance

- **Events Over Time By Protocols**
  - Legend:
    - nfs4
    - dcap
    - gftp
    - http
    - xrootd
    - restore tape
    - restore sftp
    - store tape
    - store sftp

  - Bar chart showing data over time.

- **Total Bytes Read**
  - Count: 6,676
  - Sum of transfer_size: 2.108TB

- **Total Bytes Write**
  - Count: 1,066
  - Sum of transfer_size: 256.49GB

- **Top 10 Pools**
  - Legend:
    - cms:store@osm
    - atlas:atlaslocal/groupdis...
    - cms:snuser@osm
    - dcache:replica@osm
    - dcache:replica@osm

- **Top 10 Clients**
  - Legend:
    - 131.169.16...
    - 131.169.71...
    - 131.169.16...
    - 131.164.20...
    - 134.61.26.92

- **Top 5 Storage...**
  - Legend:
    - cms:store@osm
    - atlas:atlaslocal/groupdis...
    - cms:snuser@osm
    - dcache:replica@osm
    - dcache:replica@osm

- **Efficiency Over Time**
  - Legend:
    - error_code=0
    - error_code>0

  - Line chart showing efficiency data over time.

- **Bytes In / Out by Protocols**
  - Legend:
    - nfs4
    - dcap
    - gftp
    - http
    - xrootd
    - restore tape
    - restore sftp
    - store tape
    - store sftp

  - Line graph showing bytes in/out over time.
Too good to be the truth

> Elasticsearch is very giddy component

> Number of active indexes is limited by file descriptors
  - #indexes * #shards * #replicas * #segments
  - In production we can have max ~180 days

> Can't be used to analyze historic data

> But good enough for live monitors
  - not a reporting too

> Updates brake backward compatibility
  - real pain with restrictions on field names for existing documents
Development

> Iterative functional enhancements

  ▪ Each Kibana adds great functionality

> Kibana3 → Kibana4

  ▪ Different products
  ▪ Different concepts
  ▪ No migration path

> Kibana often requires new version of Elasticsearch

> Grafana – Visualization tool based on fork of Kibana
Looking back (how to organize index)

> index => "logstash-%{+YYYY.MM.dd}"
  ▪ typical search will use 1-7 indexes
  ▪ typical search data overhead one day
  ▪ limited number of indexes
  ▪ discard granularity one day

> index => "logstash-%{+YYYY.MM}" 
  ▪ typical search will use 1 index
  ▪ typical search data overhead one month
  ▪ discard granularity one month

> your 'live view' defines which type of index you need
Our infrastructure

> Elasticsearch
  - 2.0.0, 2.2.1 by end of month
  - 9 Nodes
  - Recycled hardware
  - All data replicated

> Kibana
  - 4.2.0, 4.4.2 by end of month

> Logstash

> nginx
  - OpenResty-1.9.7.3 with LuaJIT (Lua 5.1)
Our Infrastructure

Diagram showing the infrastructure with Kibana at the center, connected to various nodes labeled "site". A laptop is shown with visualizations of dCache accounting information.
Who-is-Who

> No Authentication/Authorization by default

> All data available as soon as you can get access to ES REST-API

> Shield – native commercial solution
  - Prices on request

> Different projects to solve this issue
  - Search Guard – similar to shield
  - Custom http request manipulations
Summary

> Production services produce Gigabytes of log files per day
> Crunching the millions of numbers into a useful and handy information is not a simple task.
> Modern BigData tools looks promising approach to attack the problem
> Using widely used tools let as to adopt common practices used by other communities.
Questions?