Hadoop, Fluentd cluster monitoring with Prometheus and Grafana

2016/08/26

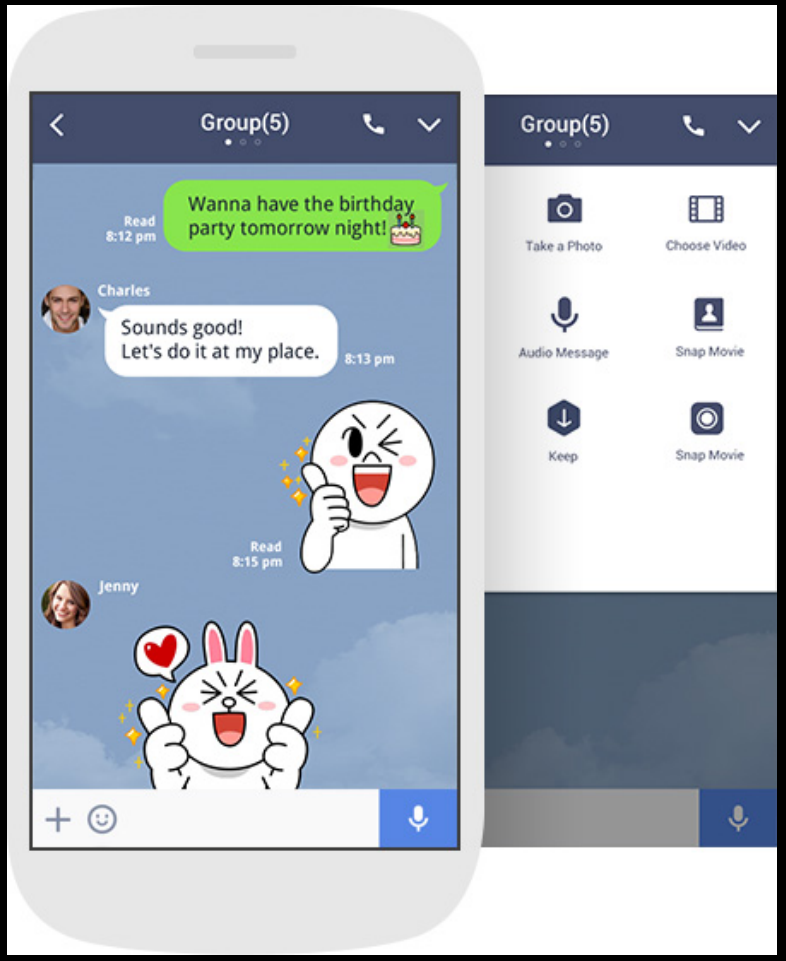
Wataru Yukawa(@wyukawa)

#promcon2016

Who am I?

- Data Engineer at LINE
- LINE makes a messaging application of the same name, in addition to other related services
- It is the most popular messaging platform in Japan

LINE


Group(5)
Read
8:12 pm
Wanna have the birthday party tomorrow night!
Charles
Sounds good!
Let's do it at my place.
8:13 pm
Read
8:15 pm
Jenny
Group(5)
Take a Photo
Choose Video
Audio Message
Snap Movie
Keep
Snap Movie

Who am I?

- First time to Germany!
- Maintain an on-premises log analysis platform on top of Hadoop/Hive/Fluentd.
- Unofficial Prometheus Evangelist in Japan
  - Organized meetup in Tokyo on June 14, 2016 (more than 100 attendences)
  - http://developers.linecorp.com/blog/?p=3908

Agenda

- Background of LINE’s development environment
- Promgen introduction
- hadoop/fluentd cluster monitoring with Prometheus and Grafana

Before Prometheus

- I have experience with other monitoring tools like Ganglia, Nagios
- I found Prometheus
  - Monitoring and alerting are unified
  - There is a query feature that allows ad-hoc queries
    - max disk usage: max by (instance) (100 - (node_filesystem_free{...} / node_filesystem_size{...}) * 100)
- I want to use Prometheus
- How do we adjust Prometheus to our environment?

LINE’s development environment

- We rarely use cloud service like AWS because we are under on-premises environment
  - host information doesn’t change frequently
- That’s why currently we don’t use any service discovery system (like Consul)
  - Therefore, we need to use static configuration for Prometheus
- We wanted to manage servers through a browser
- So, we created a tool to manage server list called **promgen** (https://github.com/line/promgen)

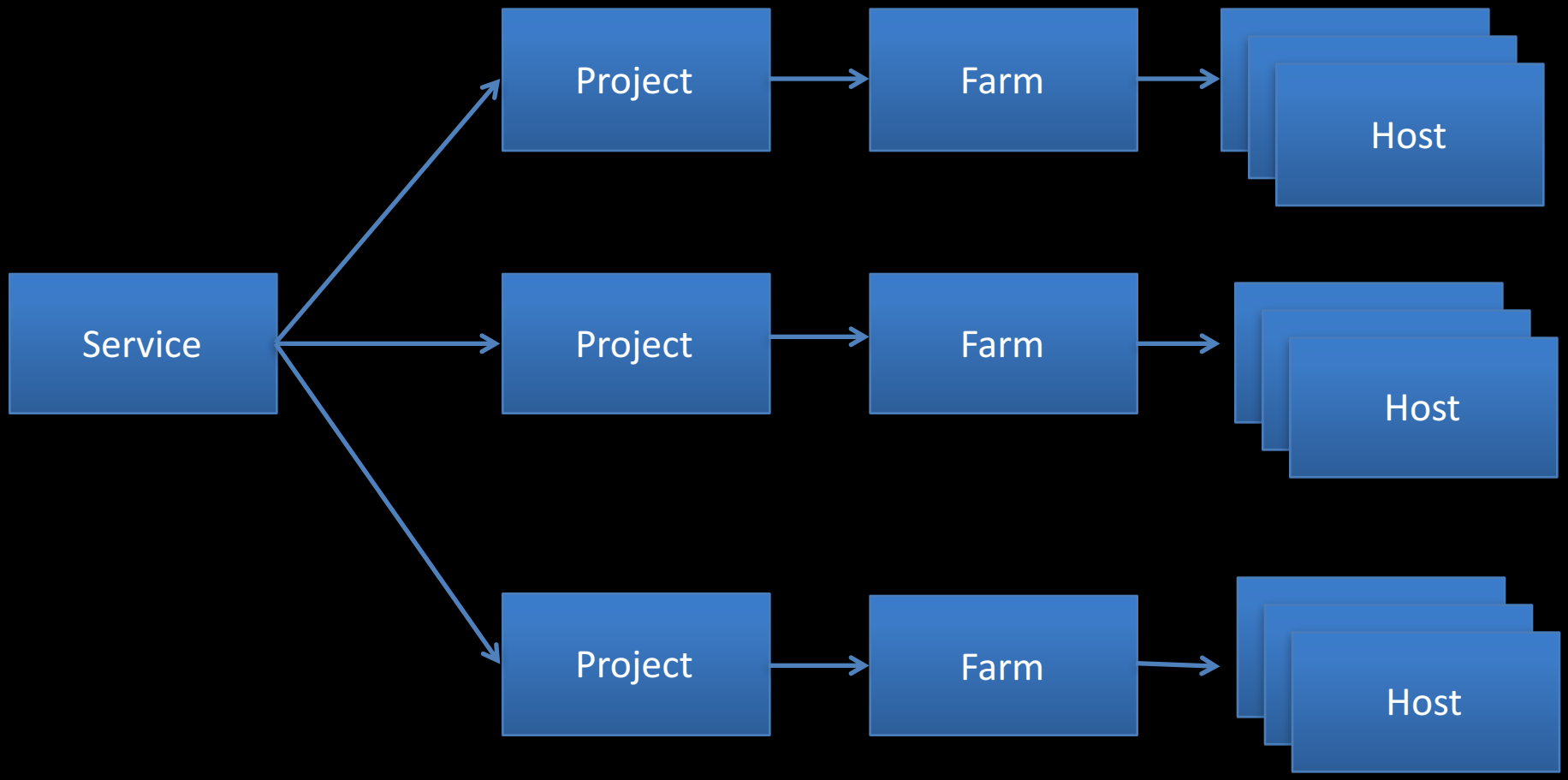
Agenda

- Background of LINE's development environment
- Promgen introduction
- hadoop/fluentd cluster monitoring with Prometheus and Grafana

About promgen

- Simple web app written in ruby which
  - Generates server list/rules and reloads(POST /-/reload) Prometheus
  - Controls alert management

Promgen data model



service of promgen

Services

Register

blog

Name	Channel	Mail Address	Webhook URL	Exporters	Farm
blog-admin	blog	a@a.localhost		node 9100 nginx 9113	blog-admin-RELEASE
blog-batch					
blog-web					

Rules | Register project | Delete service

This service has 3 projects(blog-admin, blog-batch, blog-web)

project of promgen

Home / blog-admin

alert notification

Project: blog-admin Delete

Configuration

Hipchat Channel	blog
Mail Address	a@a.localhost
Webhook URL	

Update project

Exporters

Job	Port	
node	9100	Delete
nginx	9113	Delete

Register Exporter

Hosts from blog-admin-RELEASE

Name	
blog.admin1.localhost	Delete

Add new host | Edit farm info | Unlink Farm

hosts

exporters

Host screen

Register new host to *blog-admin-RELEASE*

Name

blog.admin1.localhost
blog.admin2.localhost
blog.admin3.localhost

Paste list of host names here. Put 1 host name per line.

Register

Exporter of promgen

Home / blog-admin / Register new project exporter

Register new exporter for blog-admin

Register node_exporter(Port: 9100) Register nginx_exporter(Port: 9113)

Port

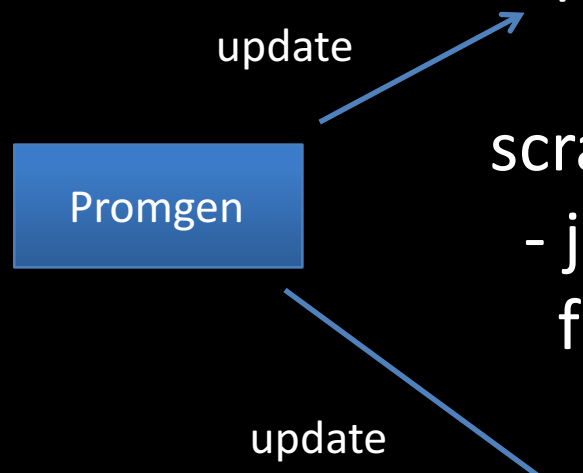
Job

nginx, apache, mysqld, node etc.

Register

Job name becomes Prometheus label

prometheus.yaml




```
rule_files:
- "/tmp/prom.rule"

scrape_configs:
  - job_name: 'dummy'
    file_sd_configs:
      - files:
        - "/tmp/prom.json"
```

Promgen doesn’t change prometheus.yaml directly,
but instead updates prom.rule/prom.json.
User should run Promgen and Prometheus on the same machine.

prom.json example

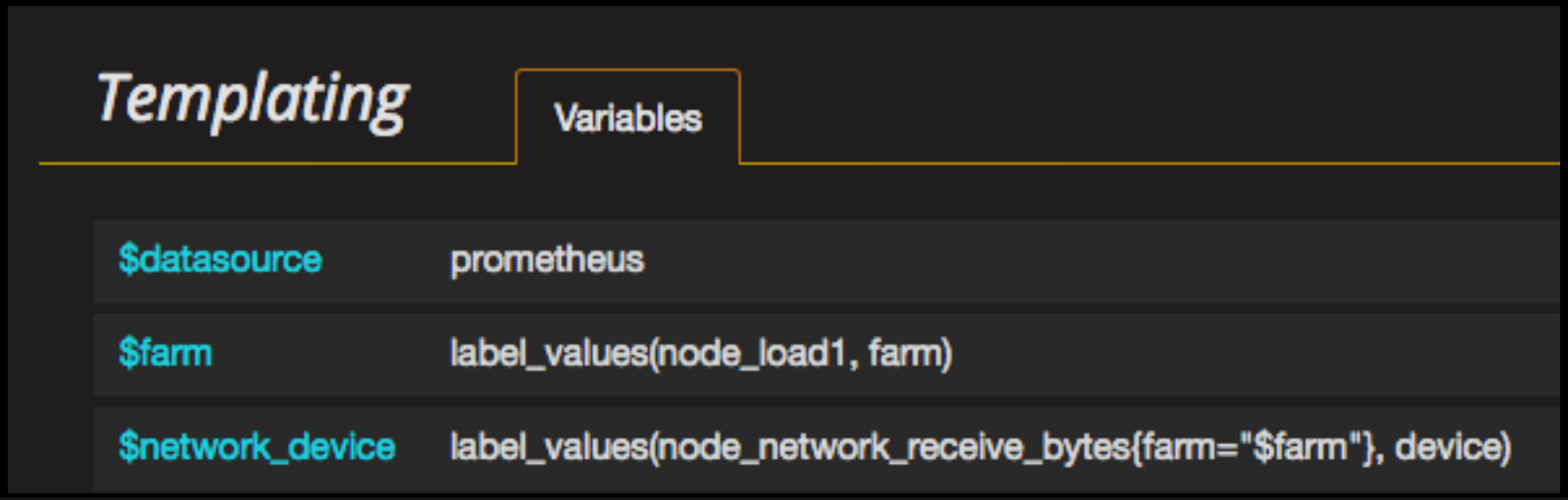
```
[
  {
    "targets":[
      "blog.admin1.localhost:9100”,
      "blog.admin2.localhost:9100”,
      ...
    ],
    "labels":{
      "service":"blog",
      "project":"blog-admin",
      "farm":"blog-admin-RELEASE",
      "job":"node"
       }
  },
...
]
```

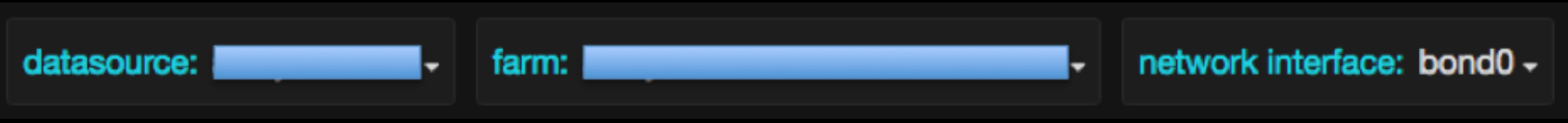
service/project/farm/job become Prometheus labels
We use this Prometheus labels in Grafana Templates

Grafana

How to use labels in templating

- Templating is useful because dashboard can be reused
- Labels correspond to templates in Grafana
- In this example, we use the farm label





Prometheus and Grafana

- Prometheus pulls metrics from exporters
- Grafana's datasouce is Prometheus
- Prometheus and Grafana are a perfect combination
- I really appreciate Grafana's Prometheus plugin

About promgen

- Simple web app written in ruby which
  - Generates server list/rule and reload(POST /-/reload) prometheus
  - Controls alert management

Alertmanager

- Alertmanager is powerful because users can avoid a flood of alert notifications
- Deduplication and silences are useful
- Alertmanager can avoid alert fatigue
- We want to manage alert notification rules and settings easily
  - for example, we want to add HipChat room and Mail address through browser.
- That’s why we implement webhook in promgen

Rule control screen

Home / blog / Register new rule

Register new rule for blog

ALERT

Alert name.

IF

Prometheus query. Example: `node_load1{...} > 5`

FOR

1 second

Prometheus to wait for a certain duration between first encountering a new expression output vector element (like an instance with a high HTTP error rate) and counting an alert as firing for this element. Elements that are active, but not firing yet, are in pending state.

LABELS

A set of additional labels to be attached to the alert. Any existing conflicting labels will be overwritten. The label values can be templated.

ANNOTATIONS

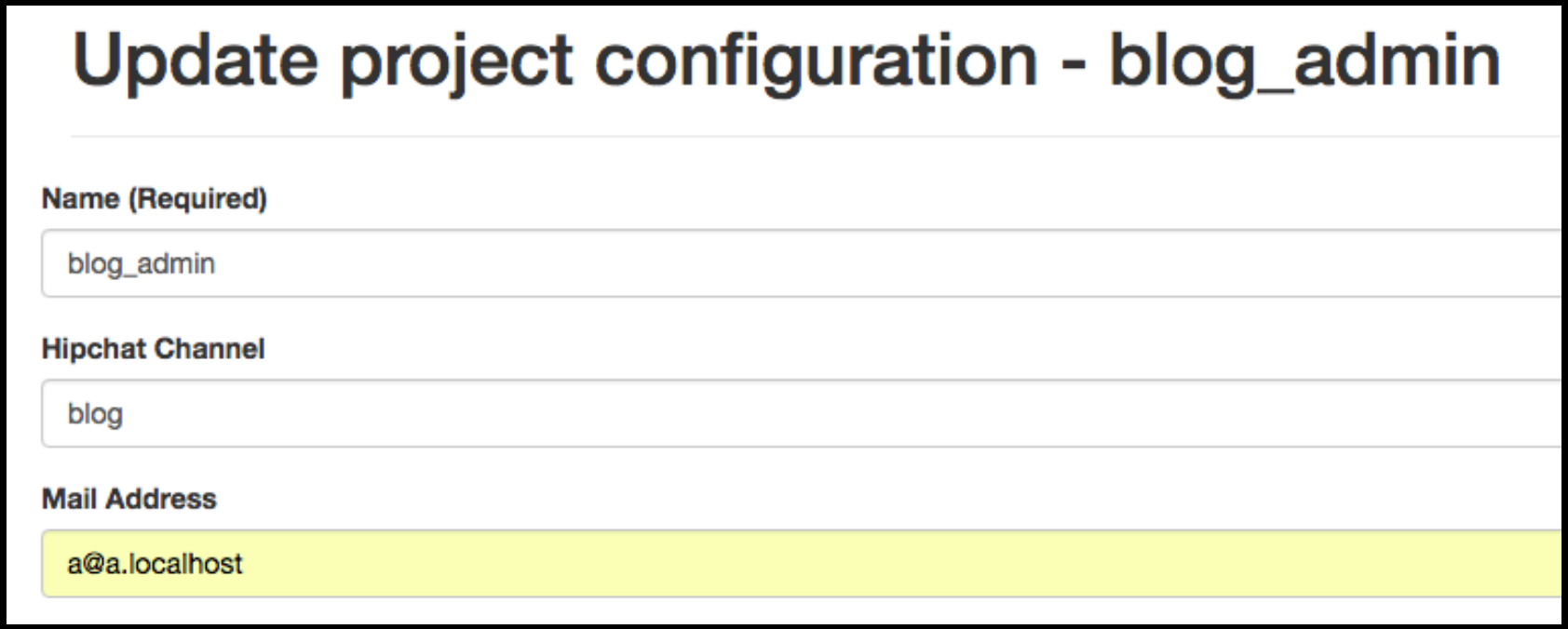
Another set of labels that are not identifying for an alert instance. They are used to store longer additional information such as alert descriptions or runbook links. The annotation values can be templated.

Register

See Alerting rules for more details.

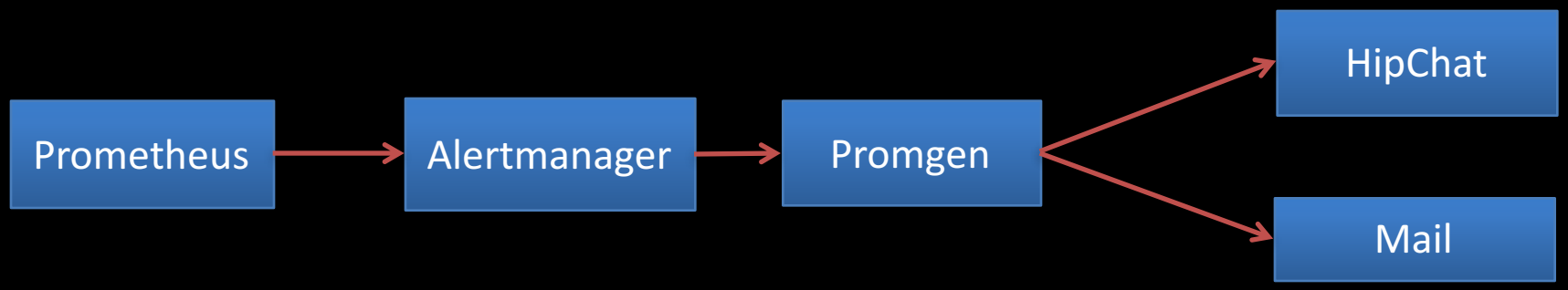
HipChat and Mail

- User can set HipChat room and mail address to receive alert



How to notify alert

- Promgen has webhook feature to send alert to both HipChat and Mail
- If alert occurs, user can receive alert through Alertmanager, Promgen

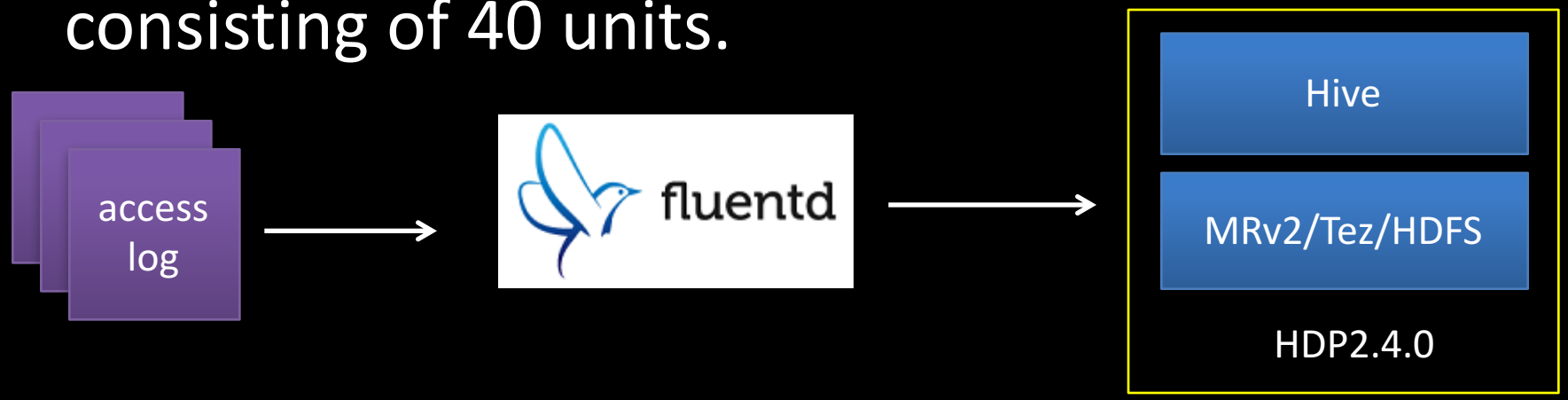


Agenda

- Background of LINE's development environment
- Promgen introduction
- hadoop/fluentd cluster monitoring with Prometheus and Grafana

Log analysis platform

- Access logs are sent to HDFS by fluentd. There are more than 400 Fluentd processes and 150kmsg/sec during peak times.
- Fluentd is an OSS log collector like logstash, flume written in ruby
- Our Hadoop cluster is medium-sized, consisting of 40 units.

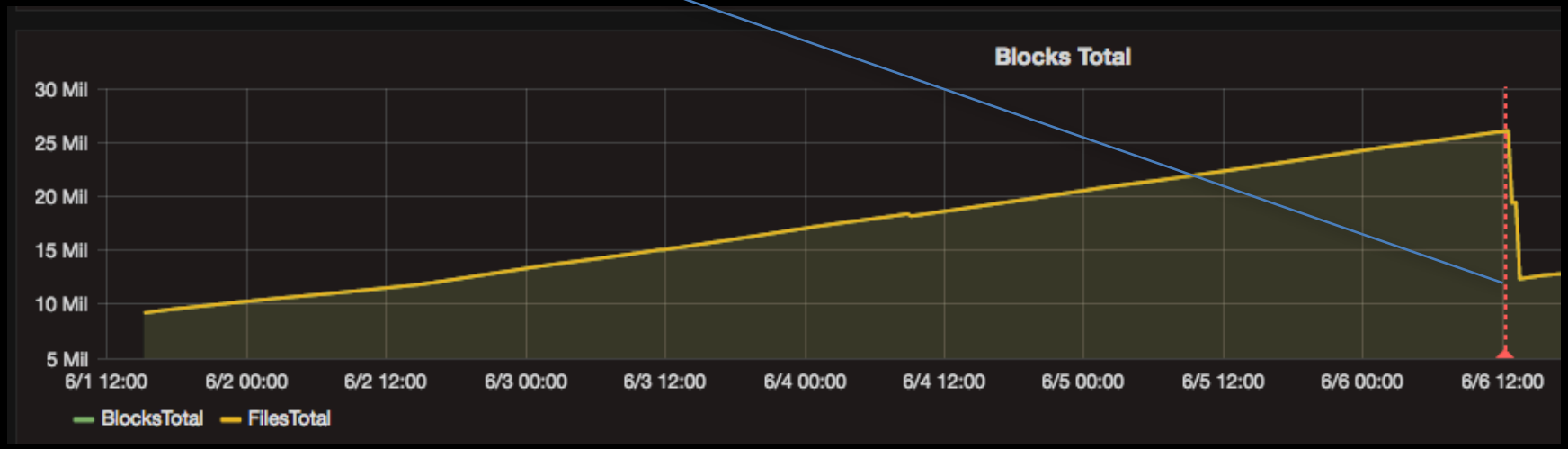


Monitoring of hadoop/hive cluster

- Developers normally use jmx_exporter to monitor java middleware
- But I wanted to create exporter, so I implemented namenode/resourcemanager/jstat exporter
- namenode_exporter uses http://namenode:50070/jmx
- resourcemanager_exporter uses http://resourcemanager:8088/ws/v1/cluster/metrics
- jstat_exporter uses jstat command
  - Honestly, current jstat_exporter implementation is not so good because when Prometheus pulls metrics, jstat command is always executed
  - cache may be necessary

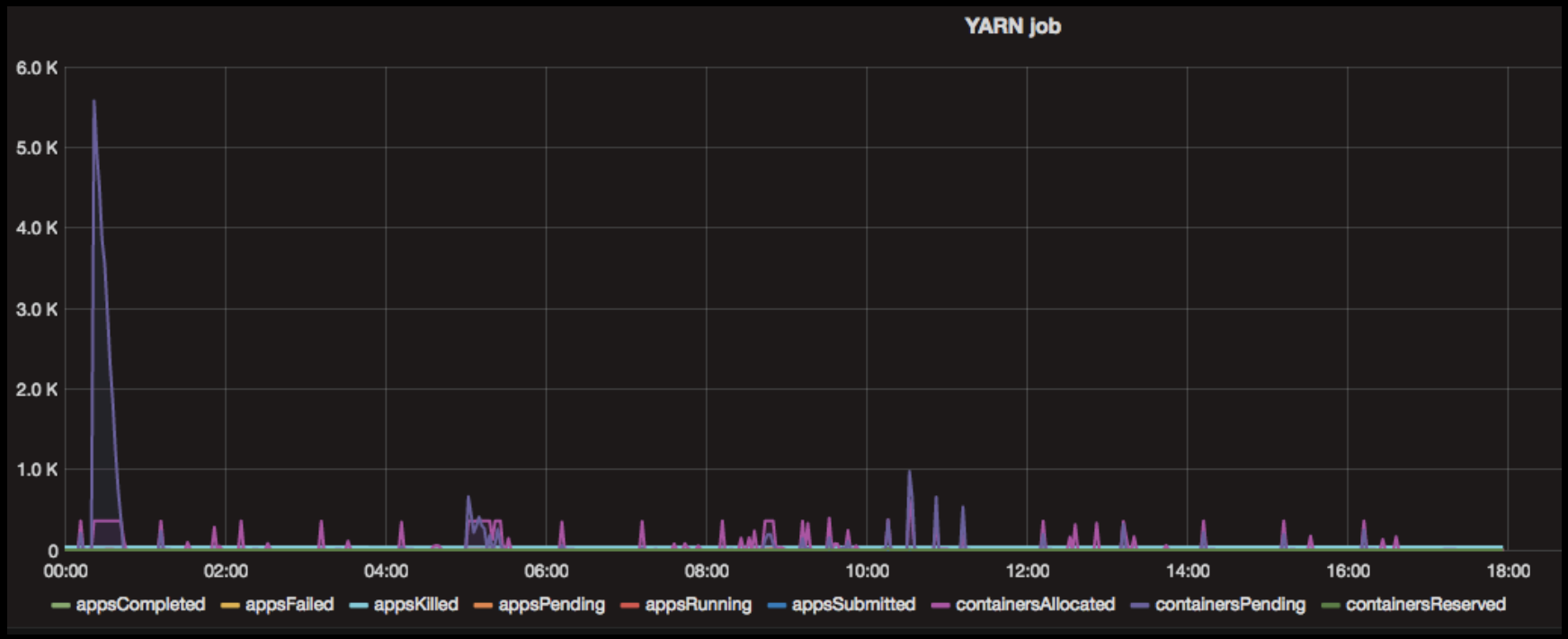
Namenode FilesTotal monitoring by using namenode_exporter

NameNode Down!



Alerts are also Prometheus metrics so Grafana can show alerts as annotations

Resoucemanager job monitoring by using resourcemanager_exporter

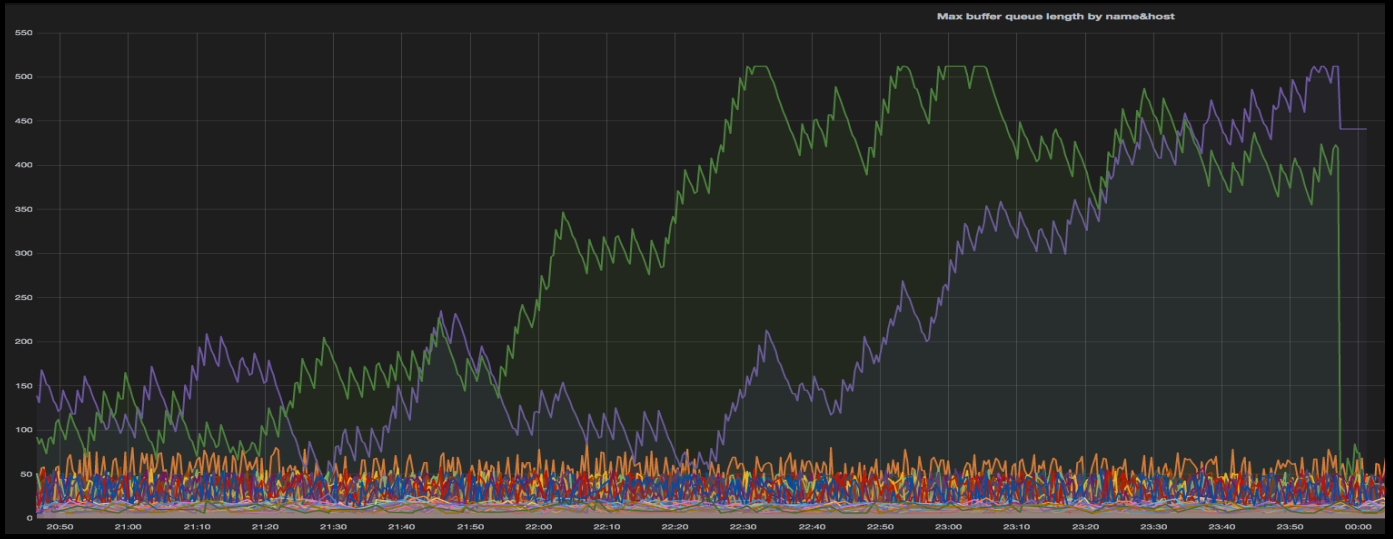


Hiveserver2 jvm monitoring by using jstat_exporter

https://issues.apache.org/jira/browse/HIVE-13374

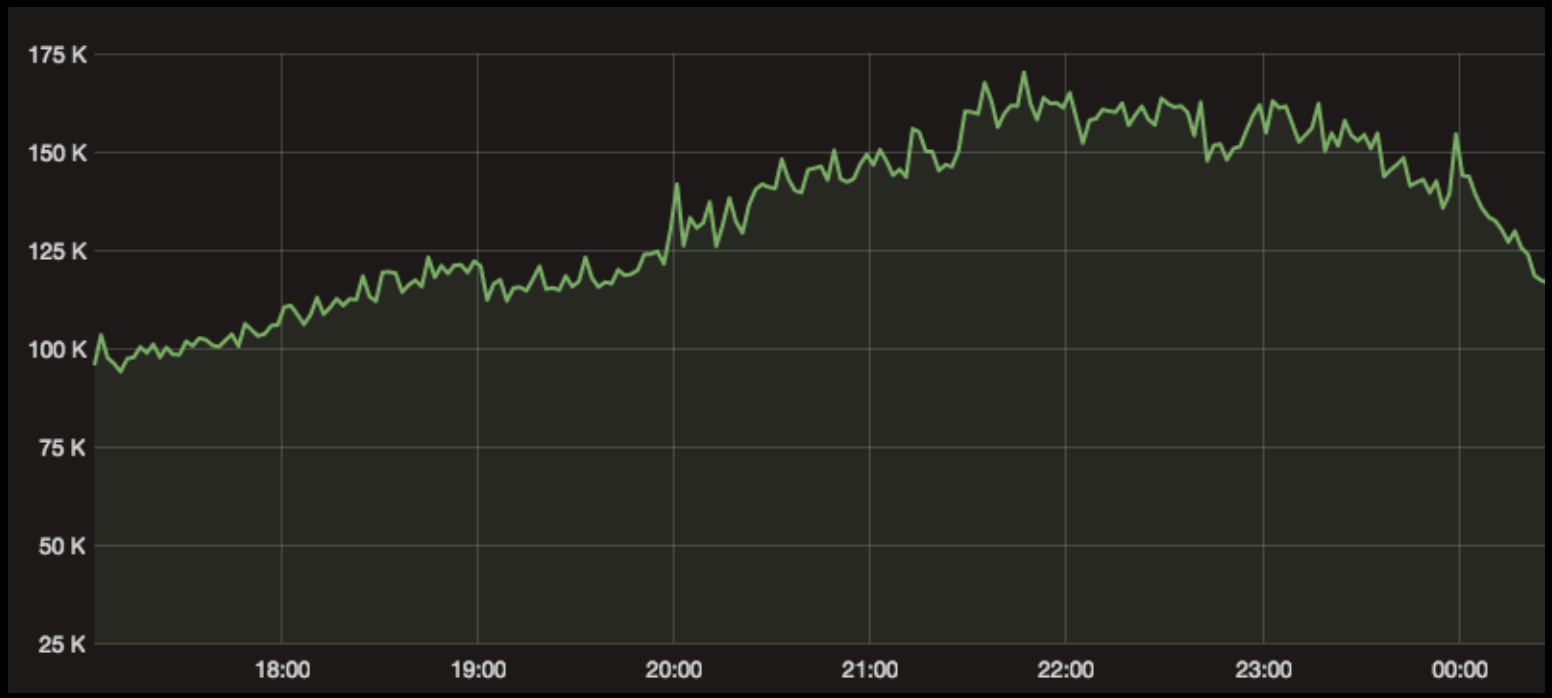
Fluentd buffer monitoring

- Fluentd has buffer mechanism to retry if destination is unstable
- fluent-plugin-prometheus enables buffer monitoring
- fluent-plugin-prometheus is fluentd plugin and use Prometheus Ruby client

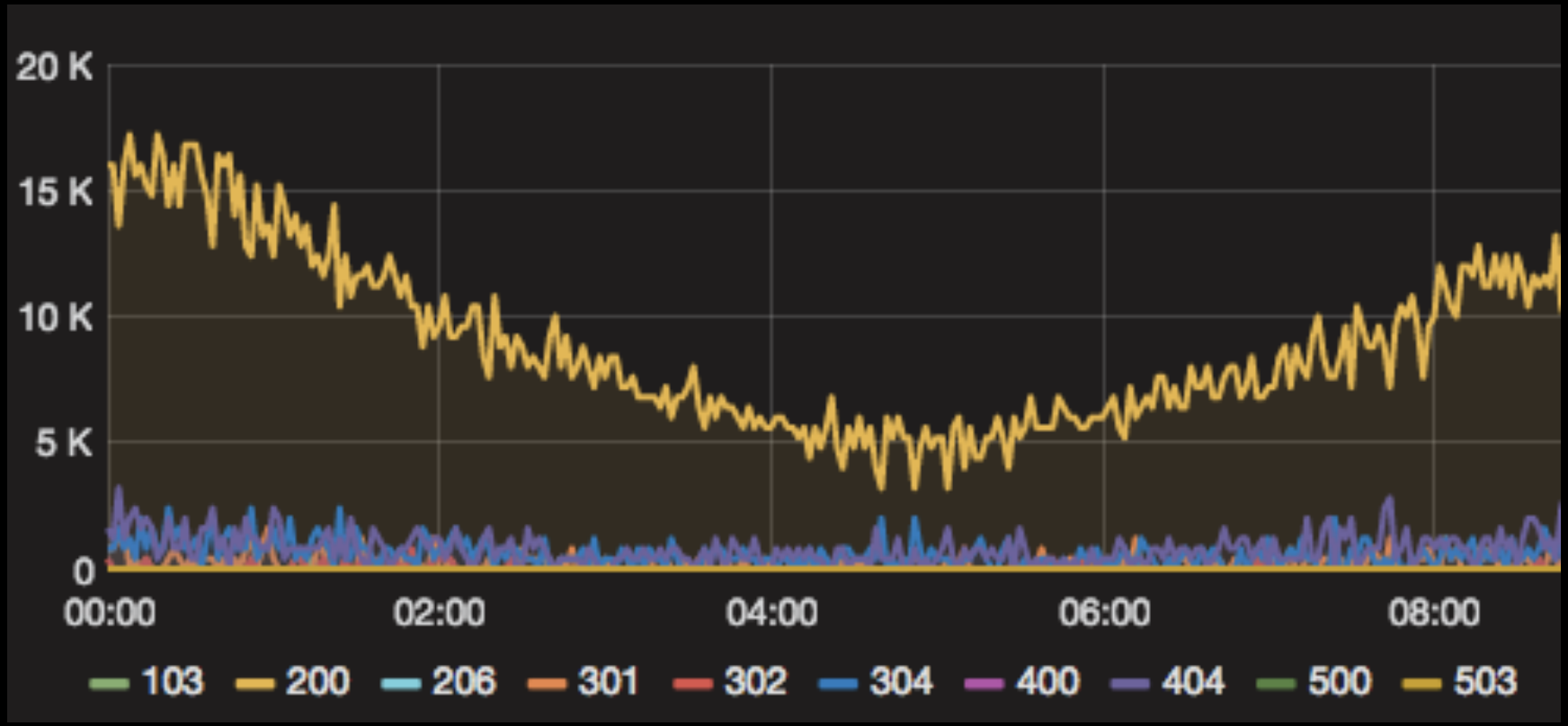


access log count

- fluent-plugin-prometheus enables us to count access log but we need sampling because of high cpu usage
- One fluentd process can‘t handle high traffic

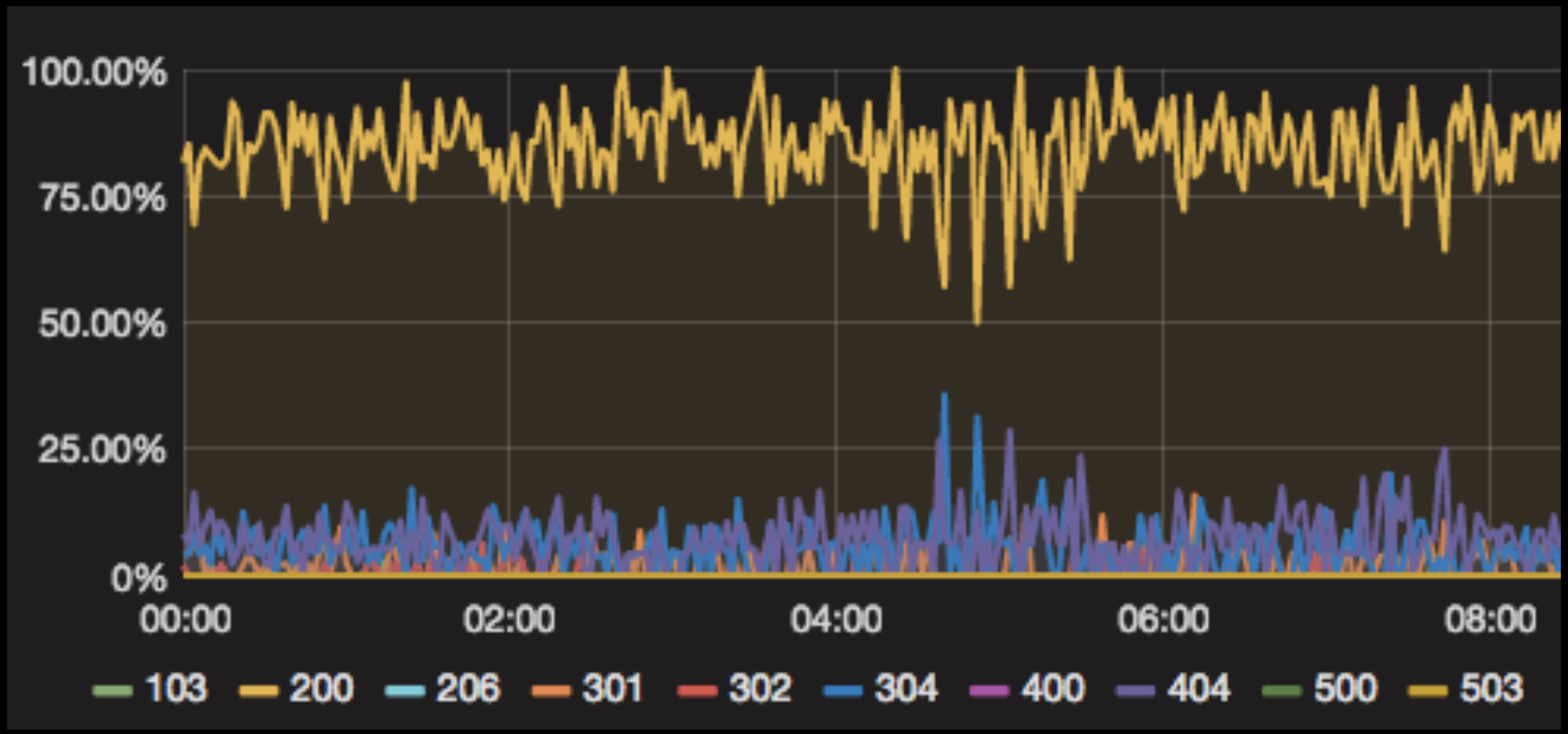


HTTP status count



Although 4xx/5xx is not 0, it may become 0 because of sampling. So we will switch to Flink.

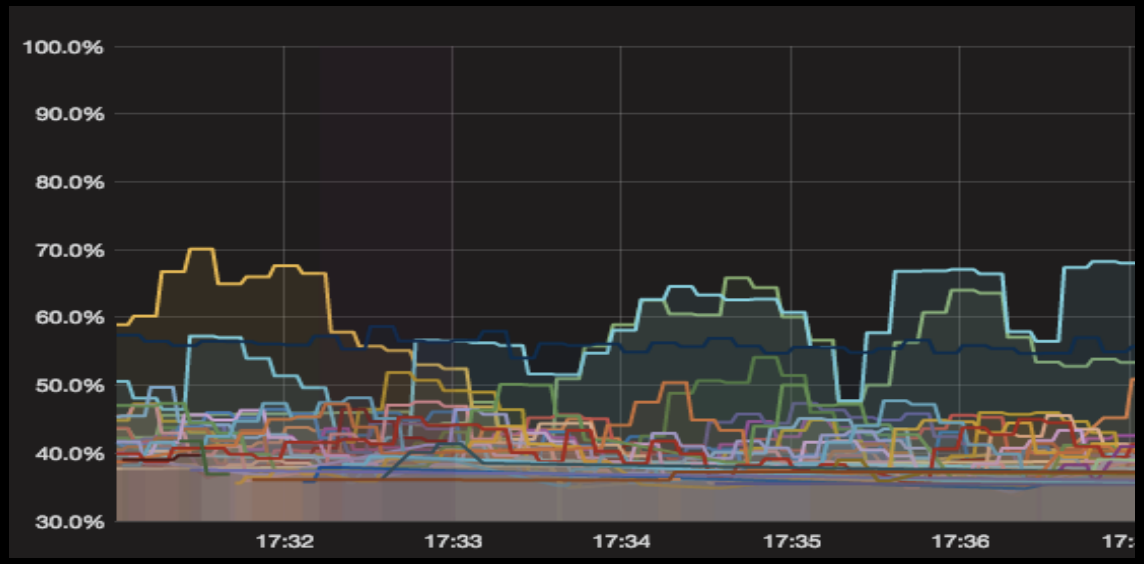
HTTP status percentage




```
sum(rate(accesslog_counts{tag="..."}[1m])) by (status, job) /
ignoring(status) group_left
sum(rate(accesslog_counts{tag="..."}[1m])) by (job)
```

fluentd_exporter

- Fluentd is often required to execute in multi process because of GVL
- I implemented fluentd_expoter to monitor fluentd cpu usage per process
- fluentd_exporter can handle multiple fluentd processes



My feeling

- Prometheus's query language is really powerful
  - sum(rate(accesslog_counts{tag="..."}[1m])) by (status, job) / ignoring(status) group_left sum(rate(accesslog_counts{tag="..."}[1m])) by (job)
- Prometheus and Grafana are a perfect combination
- We created promgen to improve host management and alert notification settings

References

- http://developers.linecorp.com/blog/?p=3908
- https://github.com/line/promgen
- http://www.fluentd.org/
- https://github.com/wyukawa/hadoop_exporter
- https://github.com/wyukawa/jstat_exporter
- https://github.com/wyukawa/fluentd_exporter
- https://github.com/kazegusuri/fluent-plugin-prometheus