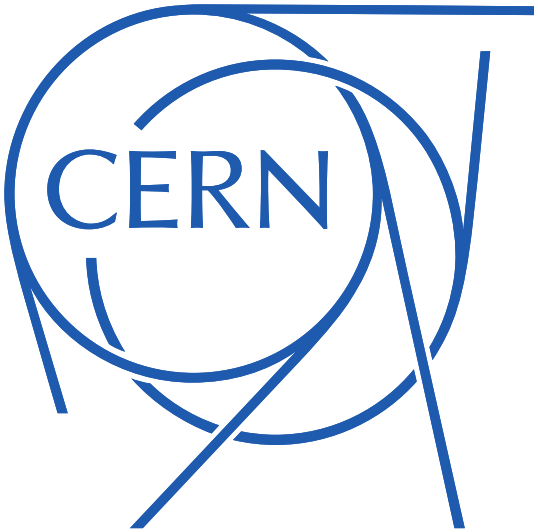

CERN

DAM-Alarming

Data Analytics from Monitoring, for alarming

Summer Student Project 2015

A. Martin, C. Cristovao, G. Domenico
thanks to Luca Magnoni

IT-SDC-MI

31/08/2015

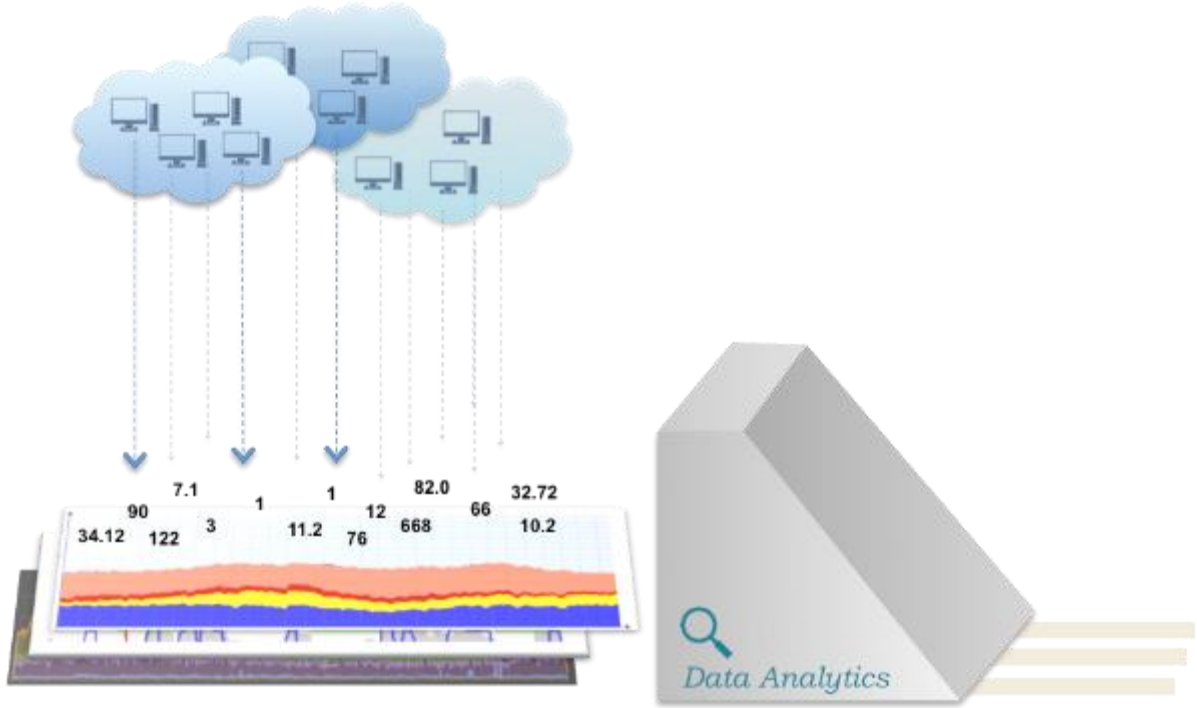
Introduction

Evaluation and implementation of CEP mechanisms
to act upon infrastructure metrics monitored by Ganglia

- WLCG is integrating cloud technologies providing an additional approach for delivering computing capacity
- Ganglia is being adopted as monitoring system for clouds
- Re-purposing raw monitoring data in order to detect anomalies

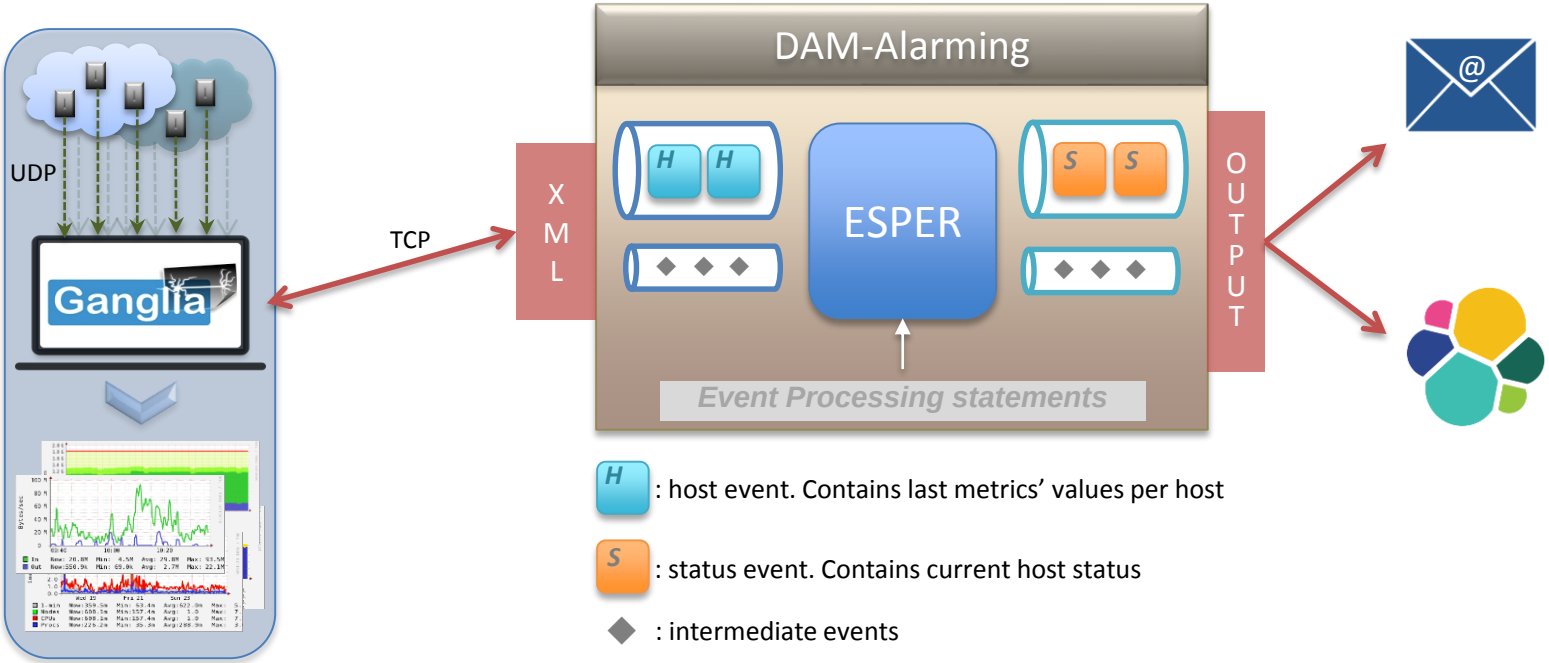
DAM Overview

- Resource profiling
- Accounting
- **Alarming**



DAM-Alarming

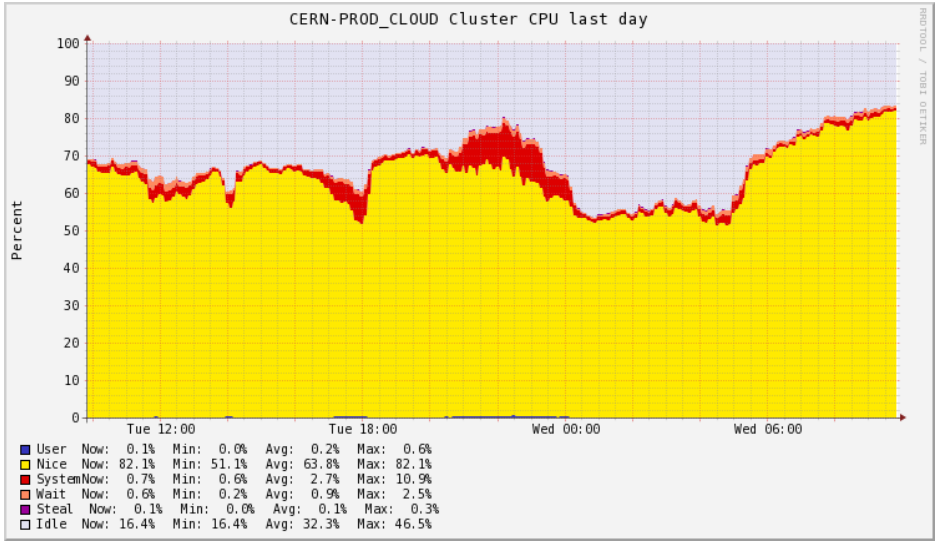
architecture



Ganglia

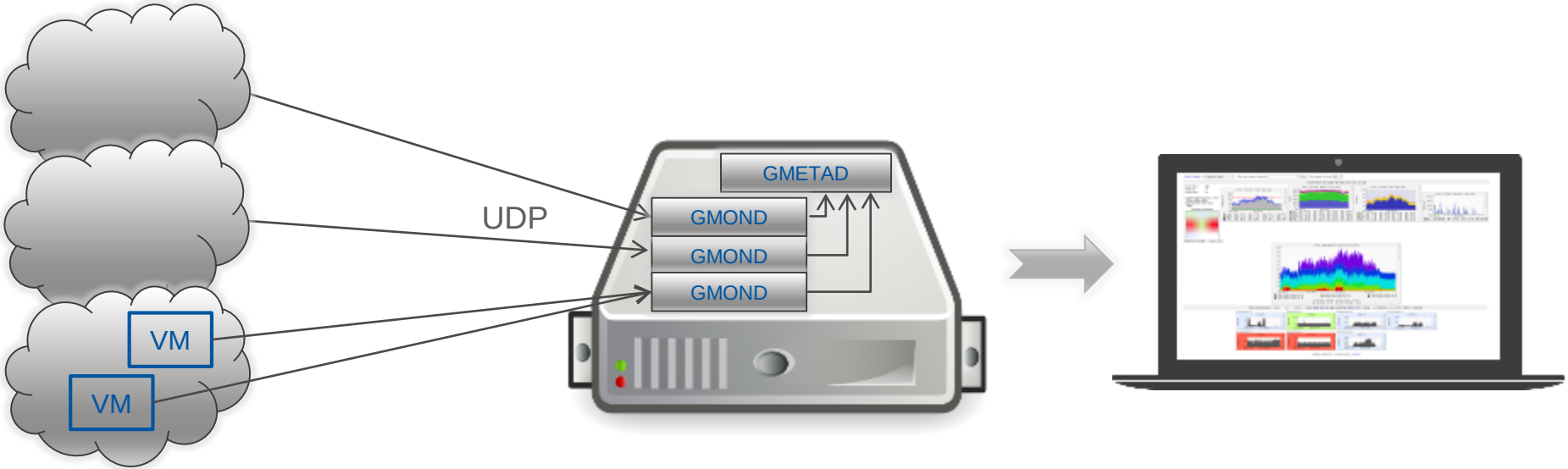
introducing the distributed monitoring system

- Scalable distributed monitoring system
- *“Designed for high-performance computing systems such as clusters and Grids”*
- Open source project
- Collecting a standard set of monitoring metrics (cpu, memory, disk, …)
- Providing web interface for visualizing host performance



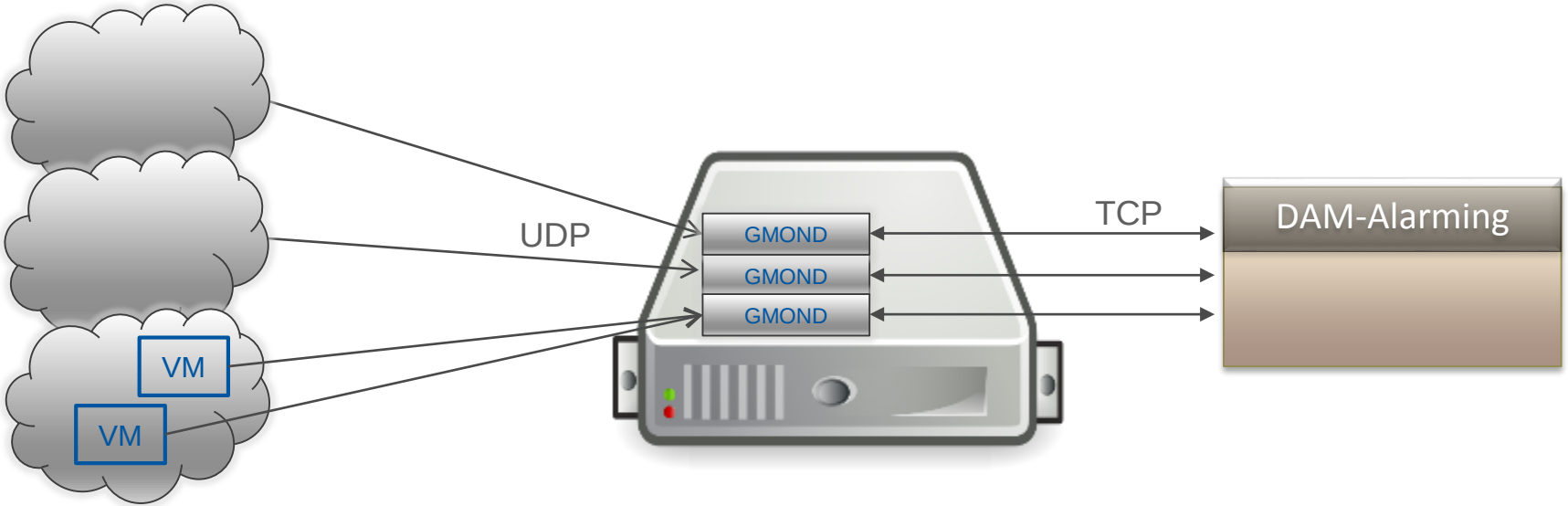
Expected deployment use case

- Multiple *gmonds* on server, one per cluster



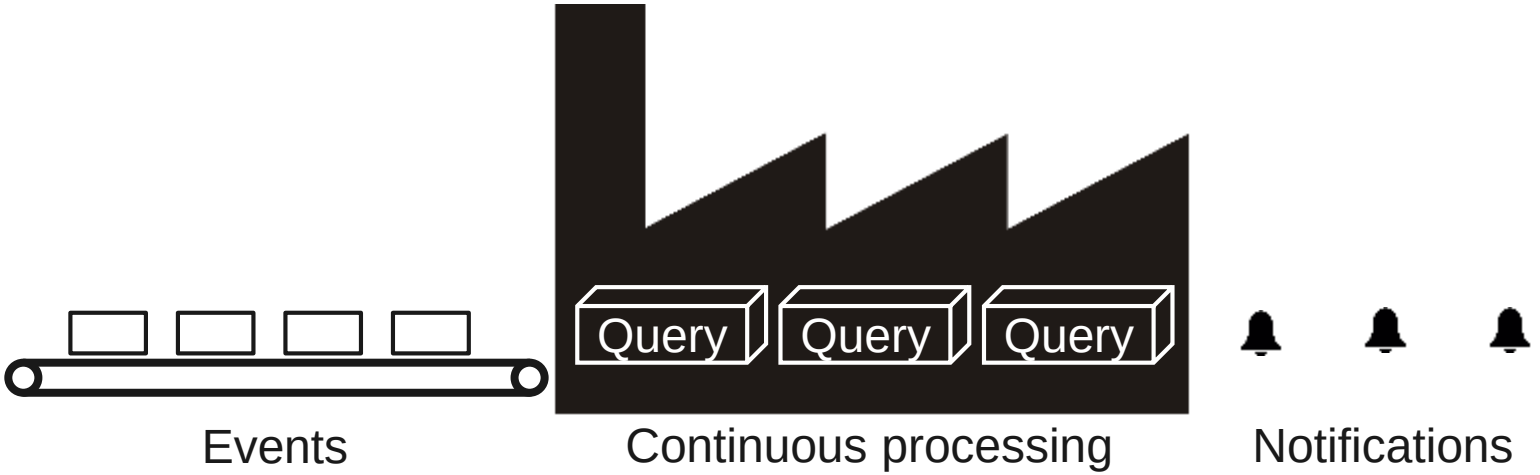
Fitting DAM-Alarming in deployment

- Contacting each *gmond* on a server to retrieve monitoring data



Complex Event Processing (CEP)

- Set of technologies to process events and discover complex patterns among their streams
- Goal: identify meaningful events and promptly respond to them



ESPER

introducing the event processing engine

Esper

- Open source CEP solution
- Strong performance, in memory computation
- Strong community and support
- In use at IT-SDC: Metis monitoring system
- Simple Java API
- Event types compliant with several input formats (including POJO and Map objects)
- Event Processing language (EPL): SQL-like statements

Examples of EPL statements

Esper

- Filtering events

```
select * from ReportEvent where cpu_idle > 85;
```

- Aggregation functions

```
select avg(cpu_idle) from ReportEvent.win:time(20 minutes);
```

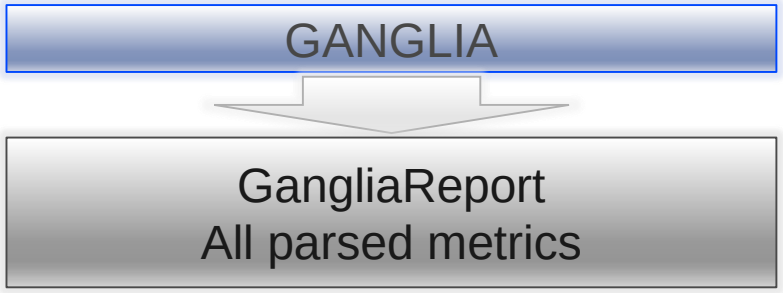
- Pattern matching

```
select * from pattern [
            every s=Status(state=State.OK) -> (
            timer:interval(5 minutes) and not
            s_update=Status(hostname=s.hostname, cluster=s.cluster))
            ];
```

Monitoring model

from events describing raw data

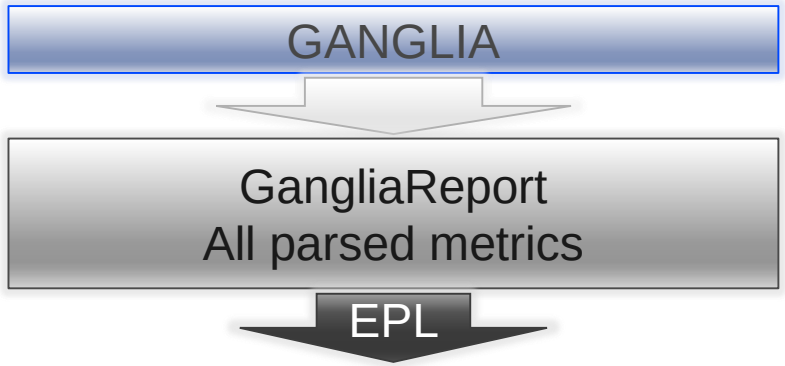
- Raw data parsed into **GangliaReport** events



Monitoring model

first layer of EPL statements

- Raw data parsed into **GangliaReport** events
- First EPL statement determines the host state based on cpu_idle
  - Possibility to correlate metrics

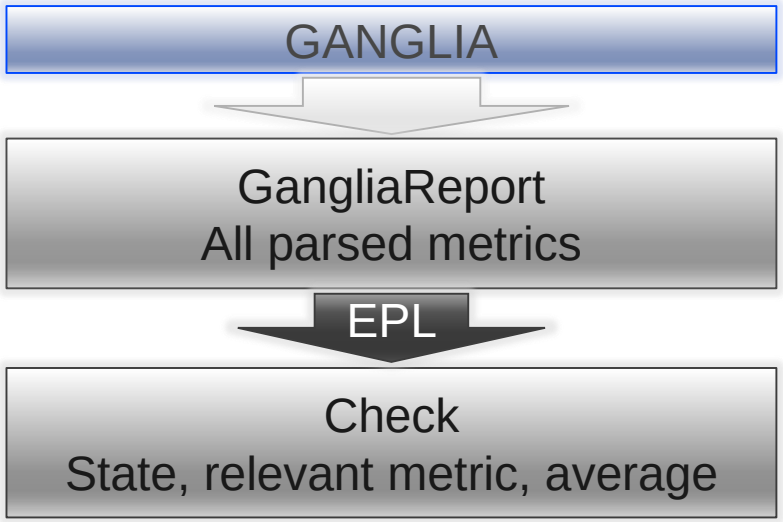



```
select      hostname, VO, cluster, monitor, reported,
            avg(cpu_idle) as ave, cpu_idle as lastVal,
            'cpu_idle' as metricName, 'high cpu_idle' as problem
            case
              when avg(cpu_idle) > 85 then State.ERROR
              when avg(cpu_idle) > 70 then State.WARNING
              when cpu_idle is null then State.UNKNOWN
              else State.OK
            end as state
from GangliaReport.win:time(var_timeWindowLength min)
group by hostname,cluster;
```

Monitoring model

events describing host state

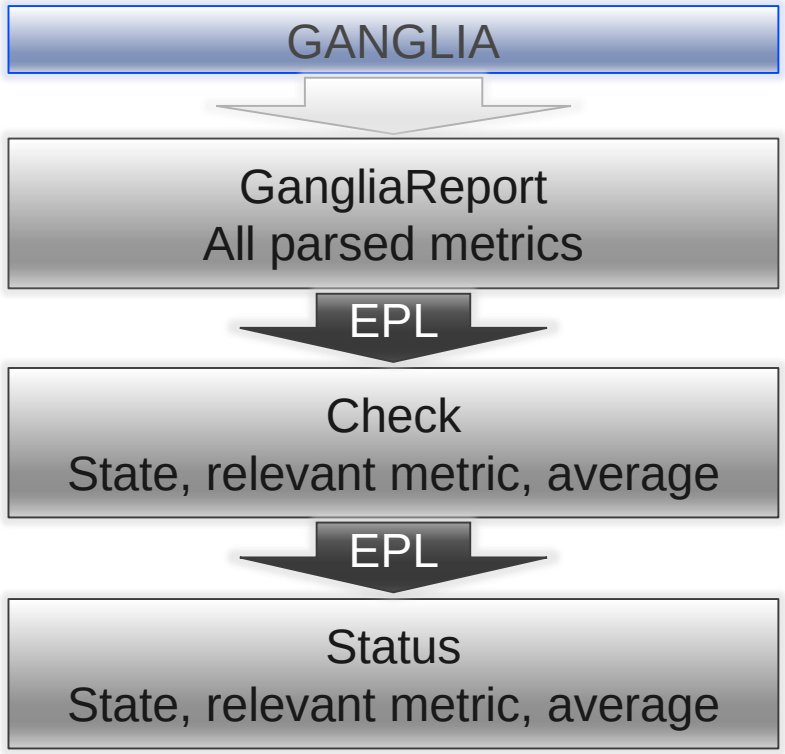
- Raw data parsed into **GangliaReport** events
- First EPL statement determines the host state based on cpu_idle
- **Check** Events describe the hosts state at current time



Monitoring model

second level of EPL statements

- Raw data parsed into **GangliaReport** events
- First EPL statement determines the host state based on cpu_idle
- **Check** Events describe the hosts state at current time
- Second EPL keeps only Check events indicating state transitions: new event **Status**

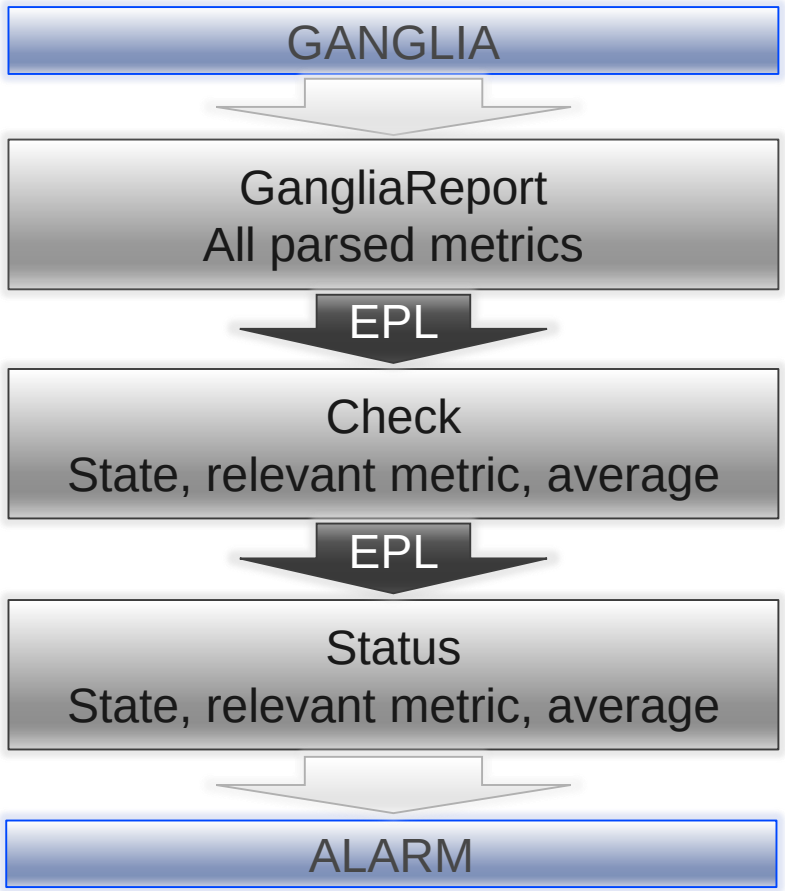



```
select * from Check.std:groupwin(hostname, cluster).win:length(2)
where state is not prev(1, state) and prev(1, state) is not null;
```

Monitoring model

alarming on status change

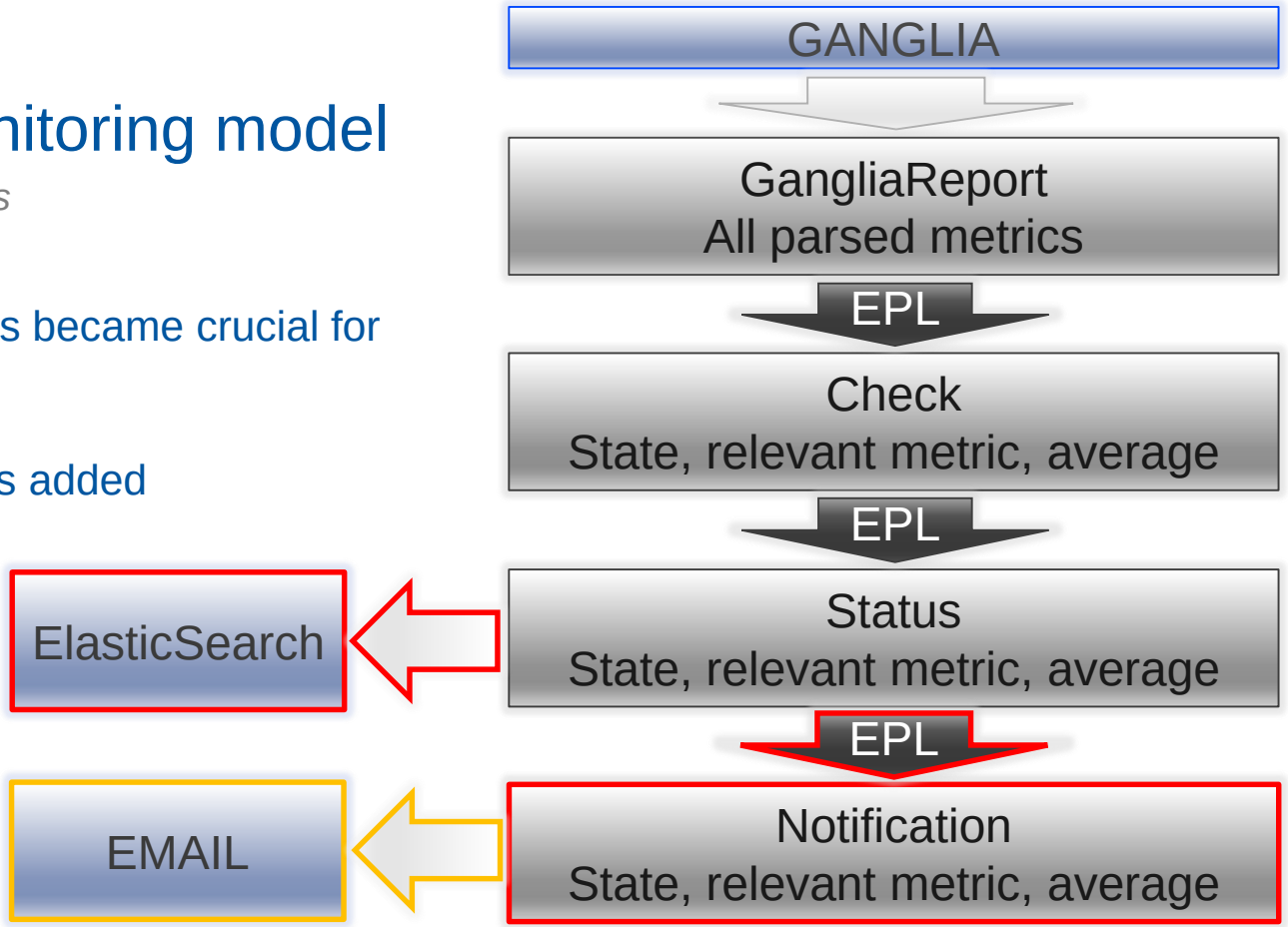
- Raw data parsed into **GangliaReport** events
- First EPL statement determines the host state based on cpu_idle
- **Check** Events describe the hosts state at current time
- Second EPL keeps only Check events indicating state transitions: new event **Status**
- State transitions trigger alarms



Upgrading monitoring model

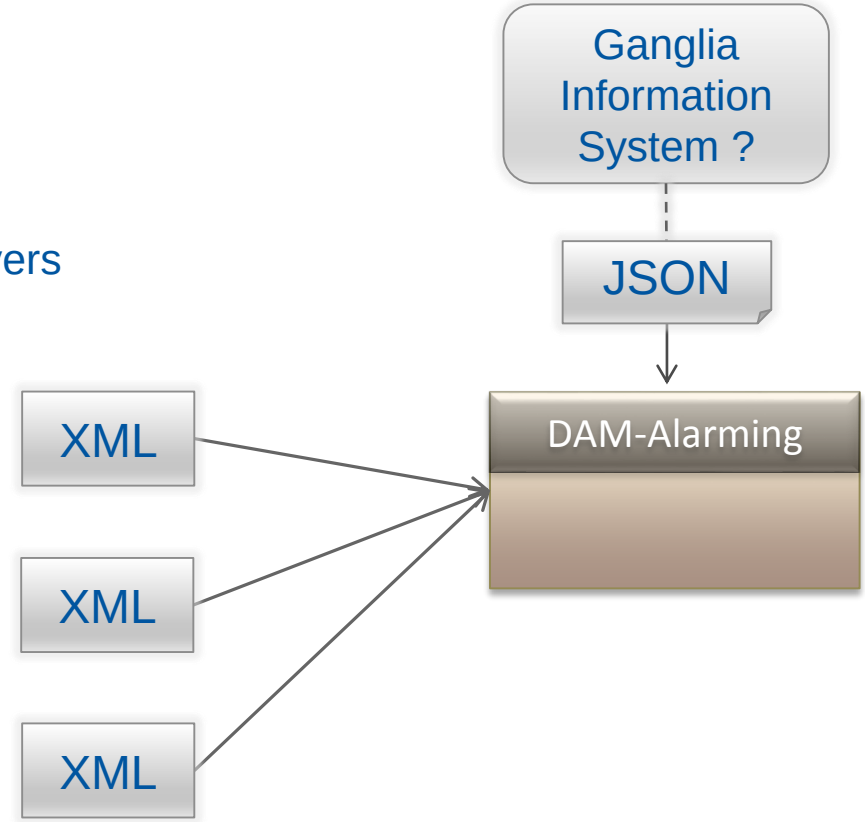
filtering email notifications

- Filtering notifications became crucial for email alarming
- Third statement was added
- All status events inserted into ES



Current deployment details

- Querying all 5 production Ganglia servers
  - Used by ATLAS, CMS and LHCb
- Poll interval: 15 seconds
- 1.5 seconds timeout for retrieving from one server



Parsing the Retrieved Data

understanding the monitoring data

snippet

```
<GANGLIA_XML VERSION="3.6.0" SOURCE="gmond">
<CLUSTER NAME="VAC.UKI-LT2-UCL-HEP.uk" LOCALTIME="1436969801" OWNER="unspecified" LATLONG="unspecified"
URL="unspecified">
<HOST NAME="lcg-wn02-02.hep.ucl.ac.uk" IP="lcg-wn02-02.hep.ucl.ac.uk" TAGS="" REPORTED="1436969544" TN="257"
TMAX="20" DMAX="1800" LOCATION="unspecified" GMOND_STARTED="1436969444">
<METRIC NAME="load_one" VAL="1.41" TYPE="float" UNITS=" " TN="267" TMAX="70" DMAX="0" SLOPE="both">
<EXTRA_DATA>
<EXTRA_ELEMENT NAME="GROUP" VAL="load"/>
<EXTRA_ELEMENT NAME="DESC" VAL="One minute load average"/>
<EXTRA_ELEMENT NAME="TITLE" VAL="One Minute Load Average"/>
</EXTRA_DATA>
</METRIC>
......                          # total of 29 metrics per host (default)
<METRIC NAME="swap_free" VAL="4194300" TYPE="float" UNITS="KB" TN="293" TMAX="180" DMAX="0" SLOPE="both">
<EXTRA_DATA>
<EXTRA_ELEMENT NAME="GROUP" VAL="memory"/>
<EXTRA_ELEMENT NAME="DESC" VAL="Amount of available swap memory"/>
<EXTRA_ELEMENT NAME="TITLE" VAL="Free Swap Space"/>
</EXTRA_DATA>
</METRIC>
</HOST>
</CLUSTER>
</GANGLIA_XML>
```

Parsing the Retrieved Data

from XML to ESPER Event

- Parsing into a Map object using SAX library
  - Metrics describing *gmond* and report (timestamp, TN, hostname, cluster,...)
  - Metrics describing host status (cpu_idle, mem_free, cpu_num, proc_total)

```
...
<HOST NAME="lcg-wn02-02.hep.ucl.ac.uk"
IP="lcg-wn02-02.hep.ucl.ac.uk" TAGS=""
REPORTED="1436969544" TN="257" TMAX="20"
DMAX="1800" LOCATION="unspecified"
GMOND_STARTED="1436969444">
<METRIC NAME="load_one" VAL="1.41"
TYPE="float" UNITS=" " TN="267" TMAX="70"
DMAX="0" SLOPE="both">
...
```

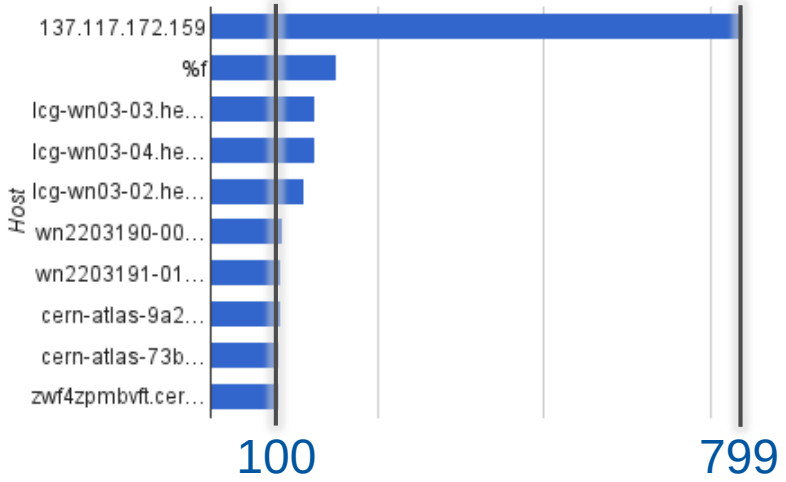
```
Map<String, Object> event = new HashMap<String, Object>();
event.put("hostname", "lcg-wn02-02.hep.ucl.ac.uk");
event.put("tn", 257);
event.put("reported", 1436969544);
event.put("cpu_idle", 85.0f);
event.put("cpu_num", 5);
event.put("proc_total", 4);
event.put("mem_free", 456123);
event.put("gmondStarted", 1436969444);
event.put("location", "unspecified");
```

First statistics from live deployment

analyzing output of first two runs connected to production servers

- First run
  - 17 hours, produced 21 000 emails, more than 1200 emails/hour
  - Averaged over 2 minutes,
- Second run
  - 16 hours, 500 email/hour
  - Averaged over 15 minutes, tweaked first statement

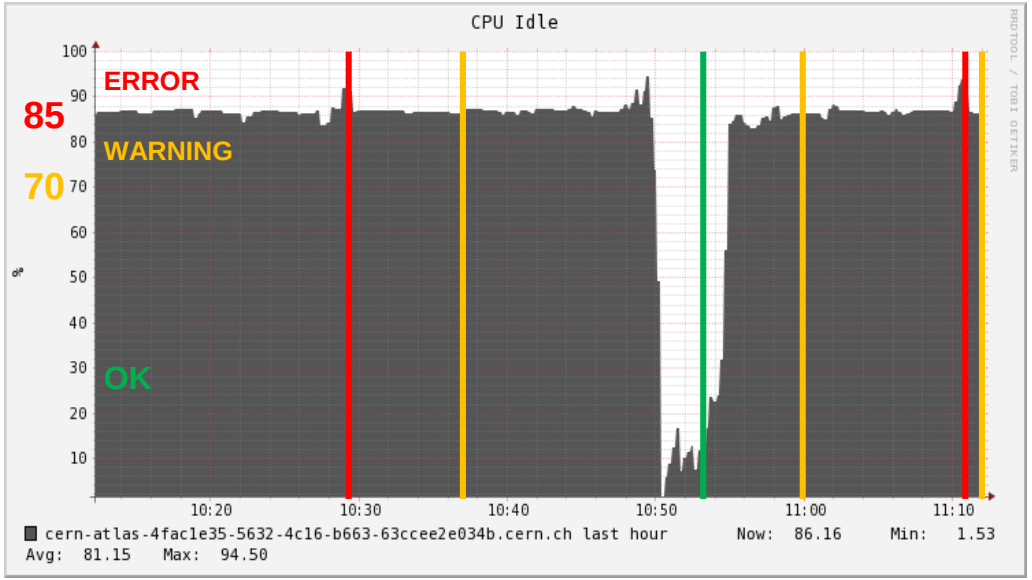
Notifications per host in first run



Filtering email notifications

hosts oscillating around fixed threshold

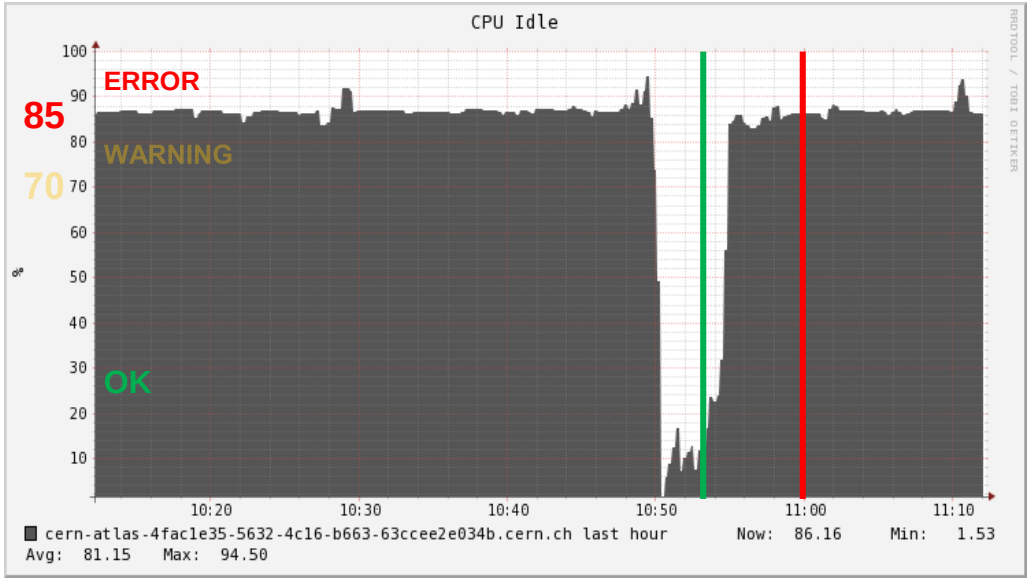
- Need to get rid of false positives
- 6 mails in 40 minutes



Filtering email notifications

hosts oscillating around fixed threshold

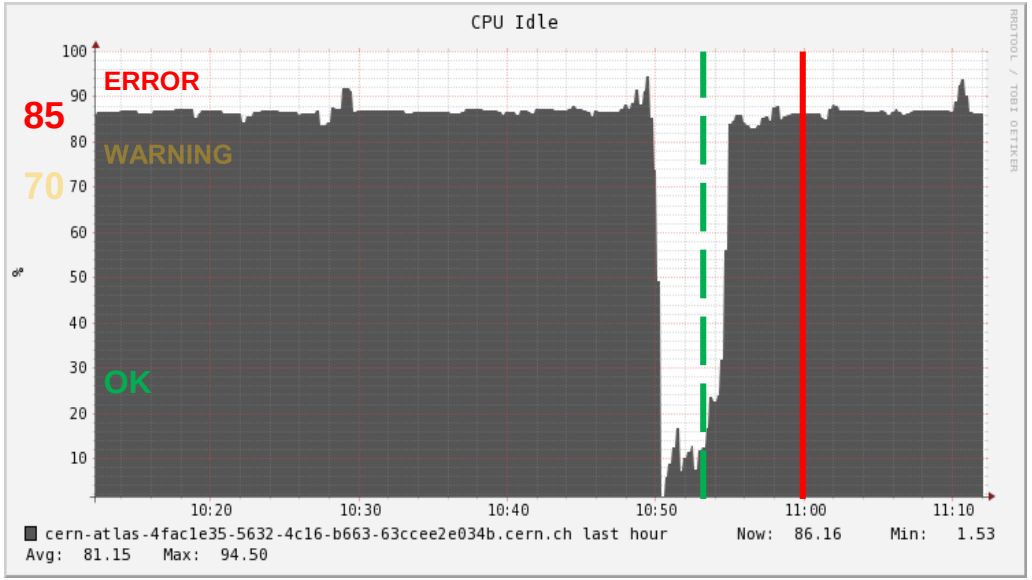
- Filtering out the notifications
- Combining states WARNING and ERROR into one



Filtering email notifications

hosts oscillating around fixed threshold

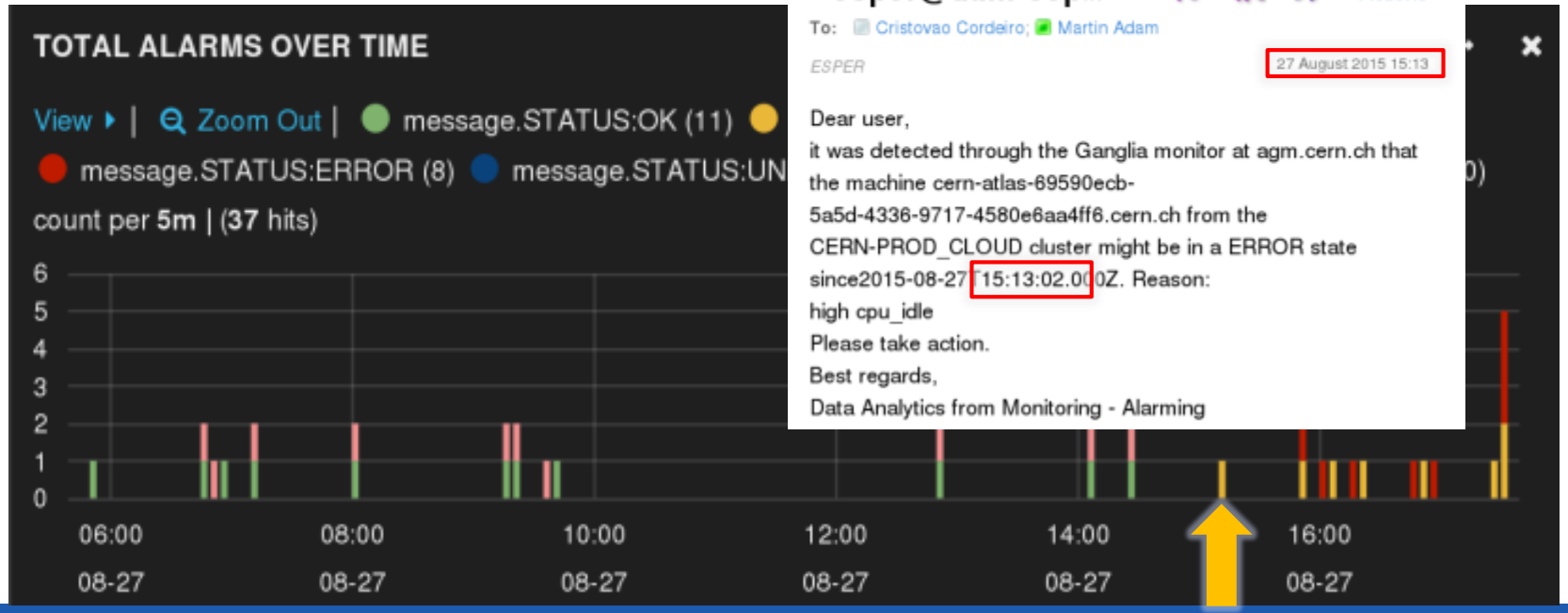
- Filtering out the notifications
- Combining states WARNING and ERROR into one
- Reporting only long lasting OK state



Filtering email notifications

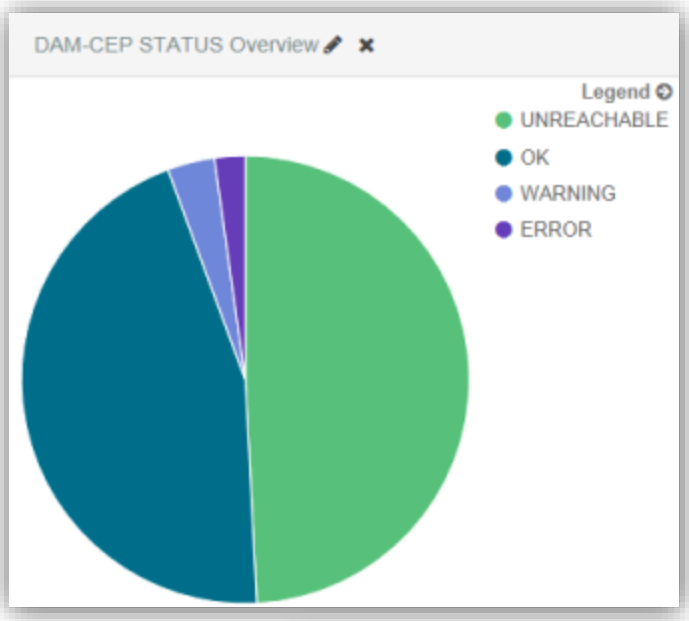
successful example

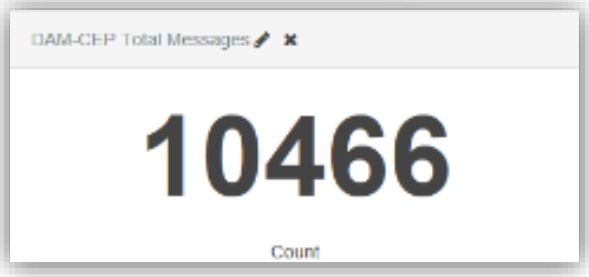
ERROR: host cern-atlas-69590ecb-5a5d-4336-9717-4580e6aa4ff6.cern.ch in CERN-PROD_CLOUD is not behaving correctly



Evaluating statistics during development

- Connected to ElasticSearch
- Sending messages via log4j to Logstash
- Visualizing using Kibana 3 and Kibana 4
- Improved statistics evaluation efficiency





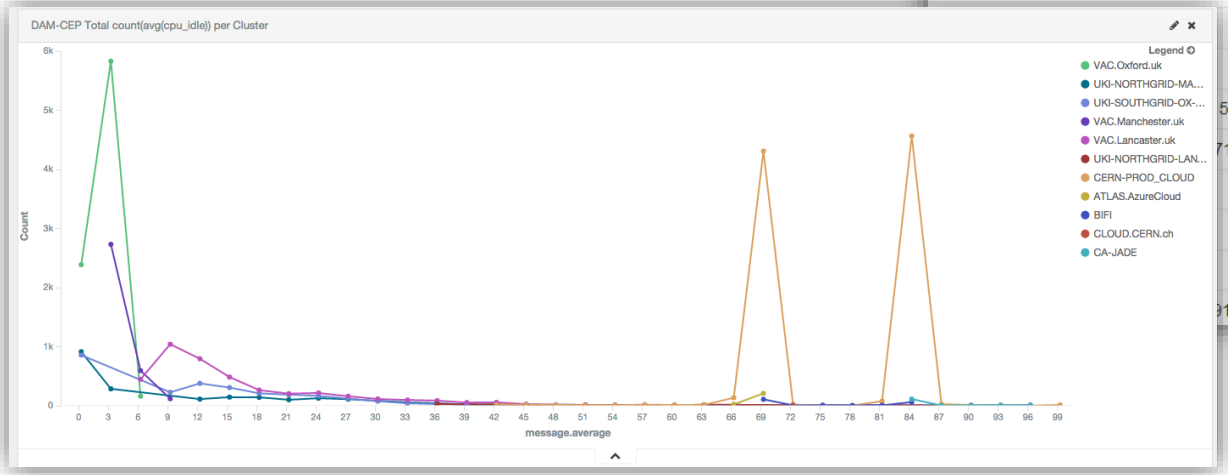
Kibana 4

more examples

- Simplifying aggregations

DAM-CEP Top 15 Hosts

Top 15 message.hostname	Count
t2vacuum04-01.physics.ox.ac.uk	402
t2vacuum04-00.physics.ox.ac.uk	352
t2vacuum04-02.physics.ox.ac.uk	271
t2vacuum03-00.physics.ox.ac.uk	142
t2vacuum04-03.physics.ox.ac.uk	142
t2vacuum11-01.physics.ox.ac.uk	115
t2vacuum07-00.physics.ox.ac.uk	111
	102
	101
5b-db5e4729398e.cern.ch	96
717-4580e6aa4ff6.cern.ch	94
	92
	83
	82
910-7c050fe2596a.cern.ch	81



Concept proven!

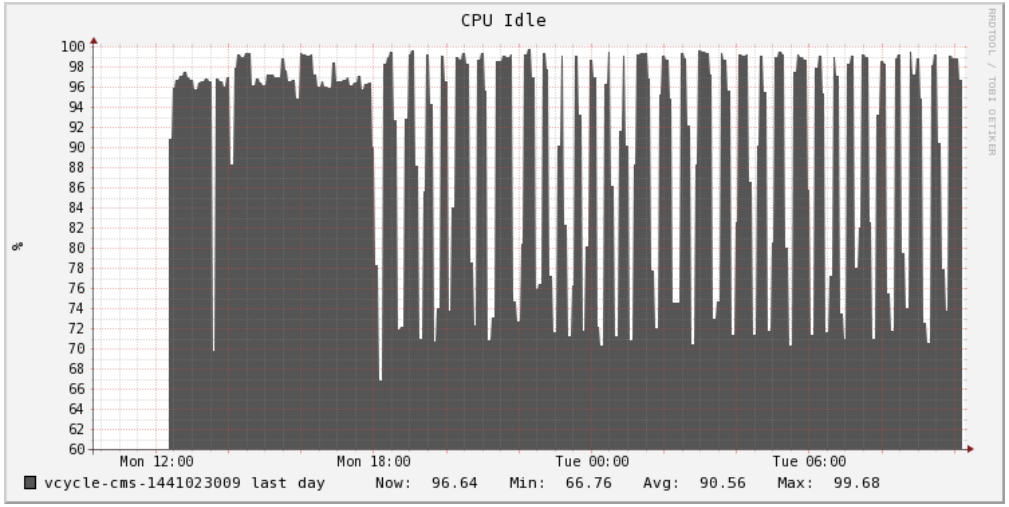
Performance example after introducing mail notification filtering

- Real run statistics:
  - Time: 27.08. 12:00 – 28.08. 12:00
  - 4417 machines
  - 8737 Status updates in ES
  - 106 emails

Challenges in alarming

hosts flapping at different rates

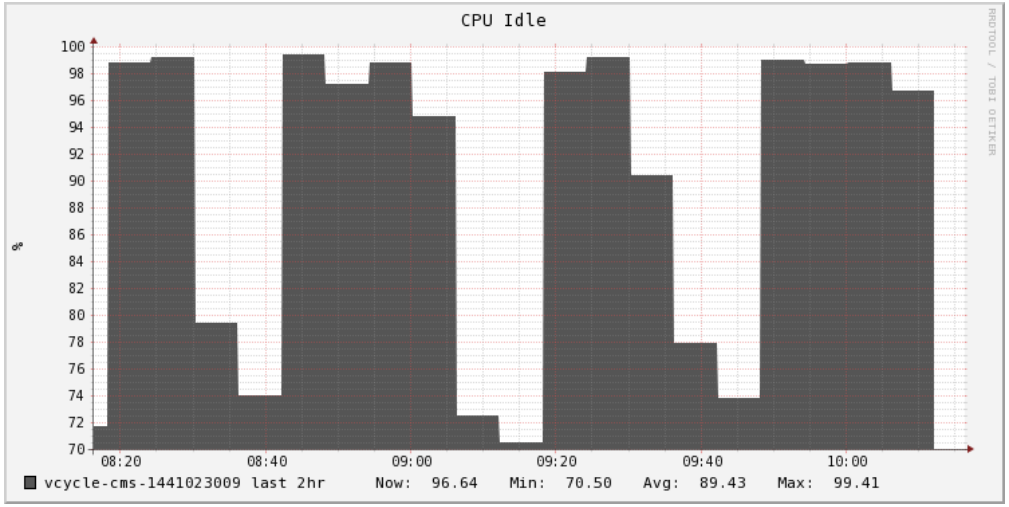
- Flapping
  - Oscillating between states
  - Difficult to detect
  - Multiple kinds
  - Variable period



Challenges in alarming

hosts flapping at different rates

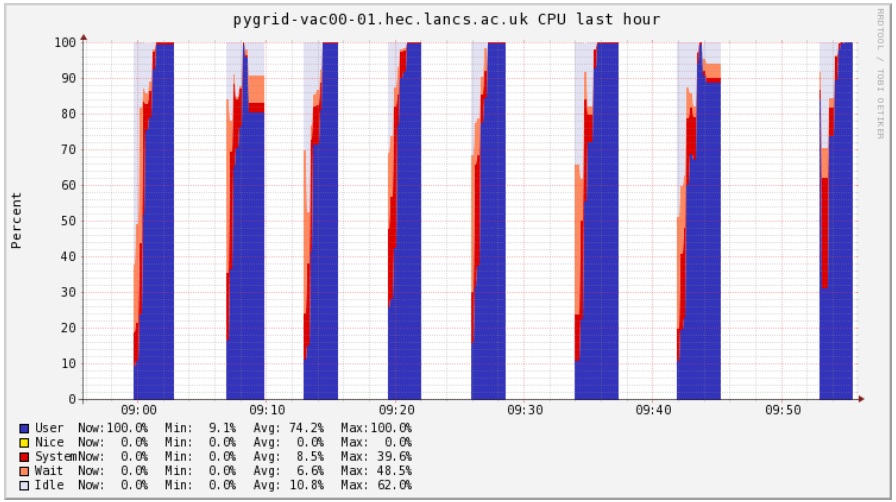
- Flapping
  - Oscillating between states
  - Difficult to detect
  - Multiple kinds
  - Variable period
- Detection based on fixed values
  - Number of status changes in a fixed time window

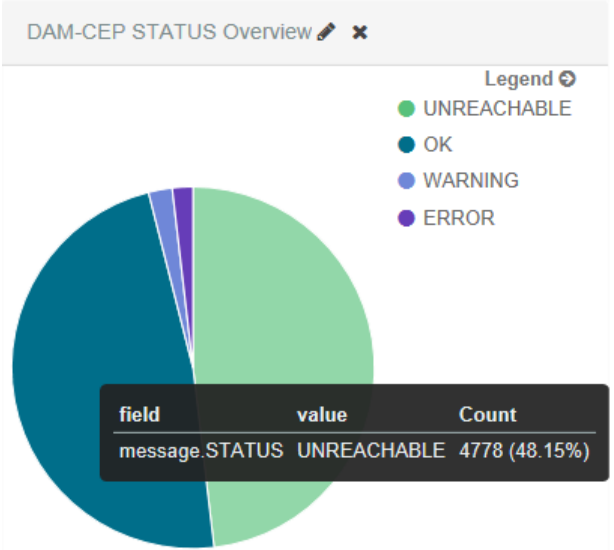


Challenges in alarming

unreachable hosts

- Detecting **Unreachable** status by checking the freshness of the report



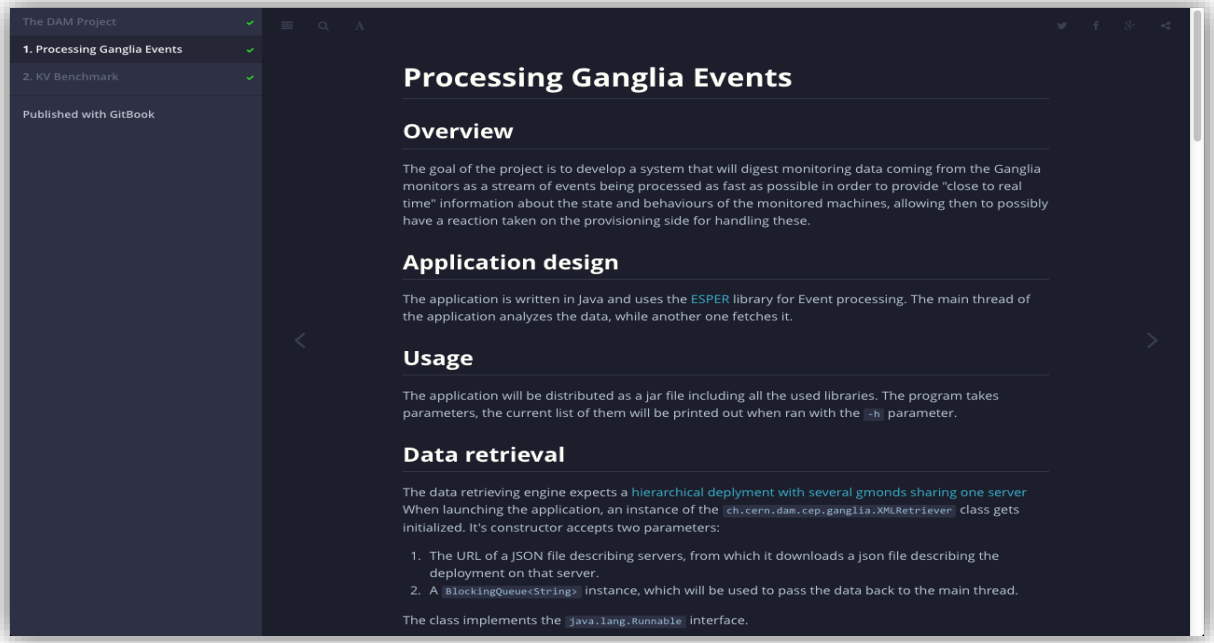


Future Work

- Improve classification
- Flapping detection
- Revision of alarming model (aggregate alarms per cluster)

Project documentation

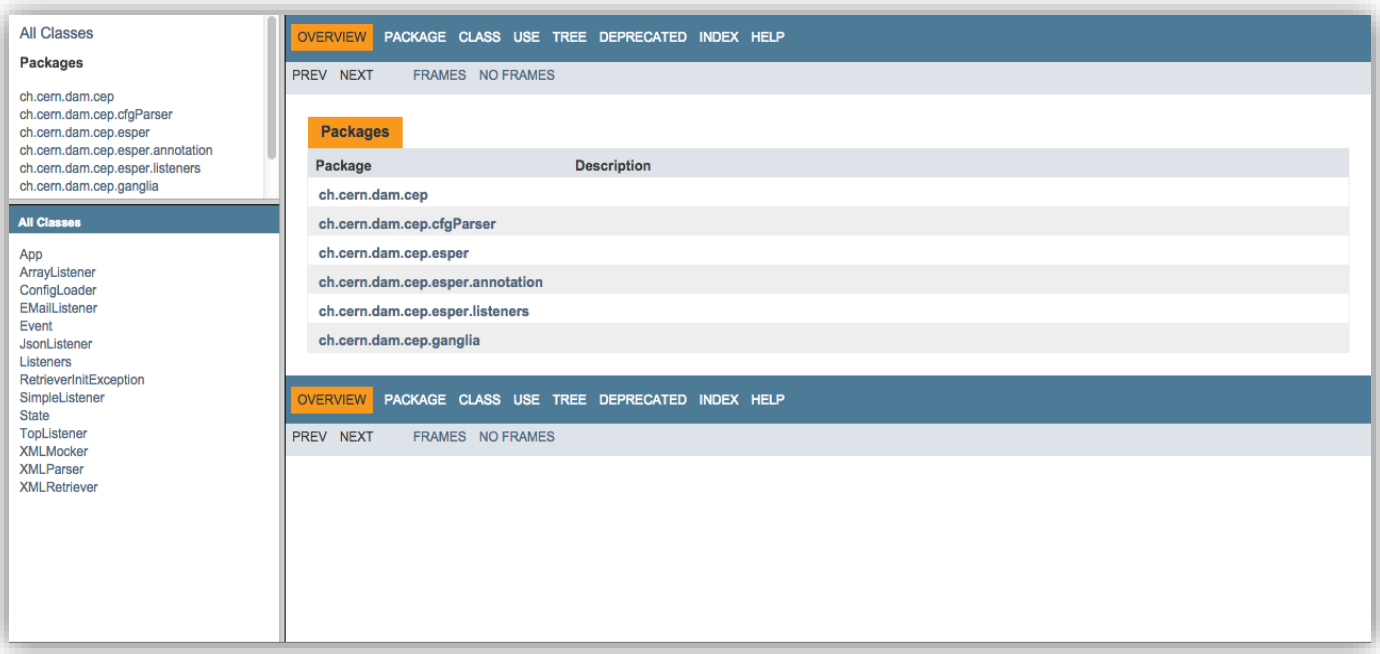
GitBook documentation with link to JavaDoc



https://sdcdam.web.cern.ch/sdcdam/DAM-CEP/index.html

Project documentation

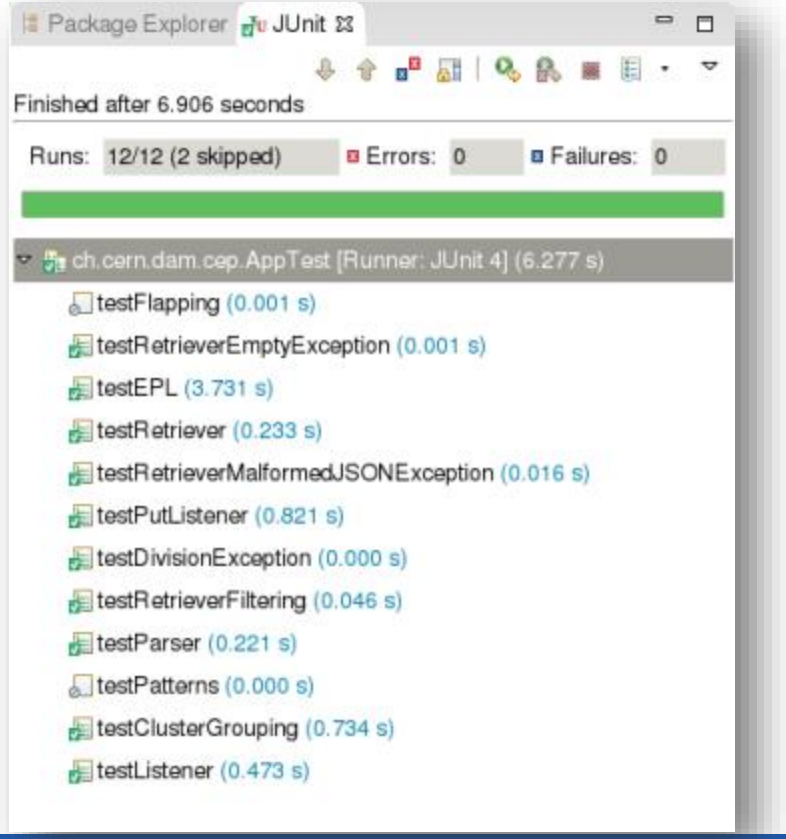
GitBook documentation with link to JavaDoc



https://sdcdam.web.cern.ch/sdcdam/DAM-CEP/javadoc/

Testing applications components

- Testing all components
- Using JUnit testing suite to implement Unit test



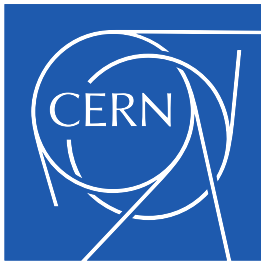
Testing EPL statements

- Using JUnit testing suite
- Help when developing statements
- Could control time to test time windows and time based patterns

```
// initializing ESPER engine and load EPL modules
EPRuntime cepRT = initEsper("test");

// creating test event
Map<String, Object> eventOK = new HashMap<String, Object>();
eventOK.put("hostname", "lhcb-cloud.cern.ch");
eventOK.put("reported", 1435924725);
eventOK.put("state", State.OK);

// sending first event
cepRT.sendEvent(eventOK, EVENT_TYPE);

// shifting time
long timeInMillis = System.currentTimeMillis();
timeInMillis += 15000;
timeEvent = new CurrentTimeEvent(timeInMillis);
cepRT.sendEvent(timeEvent);

// sending second event
cepRT.sendEvent(eventOK, EVENT_TYPE);
```

Conclusion

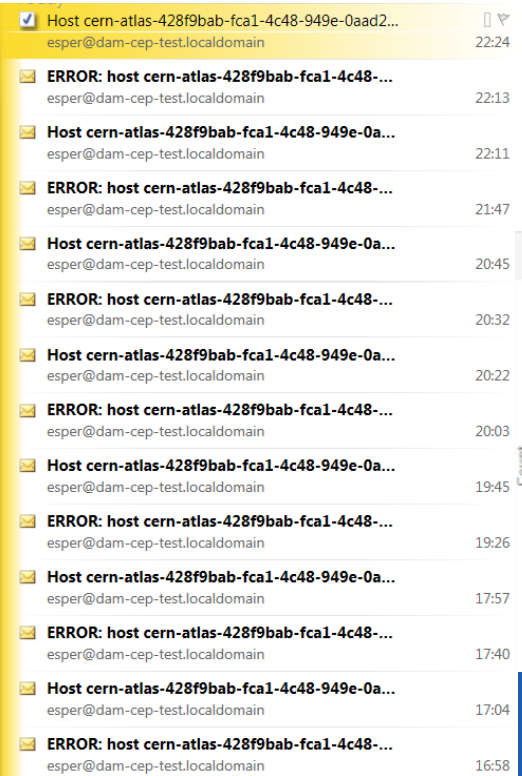
- Successfully implemented an anomaly detection and alarming system based on raw monitoring data coming from Ganglia
- Tested on 5 production cloud monitoring Ganglia servers
- Detecting anomalies based on *cpu_idle* and reporting them without spamming
- Injecting all status transitions into ES
- Processing more than 65 000 events/hour and can scale up

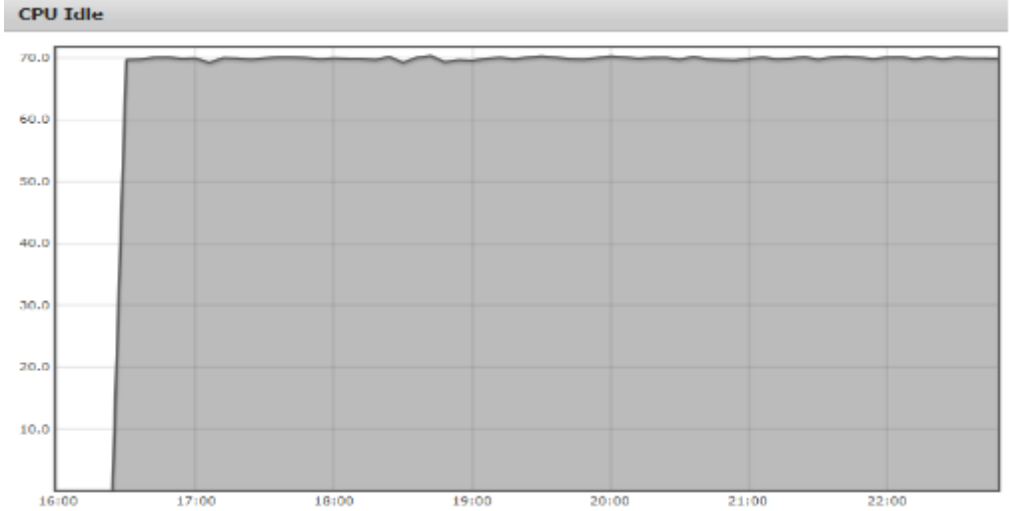


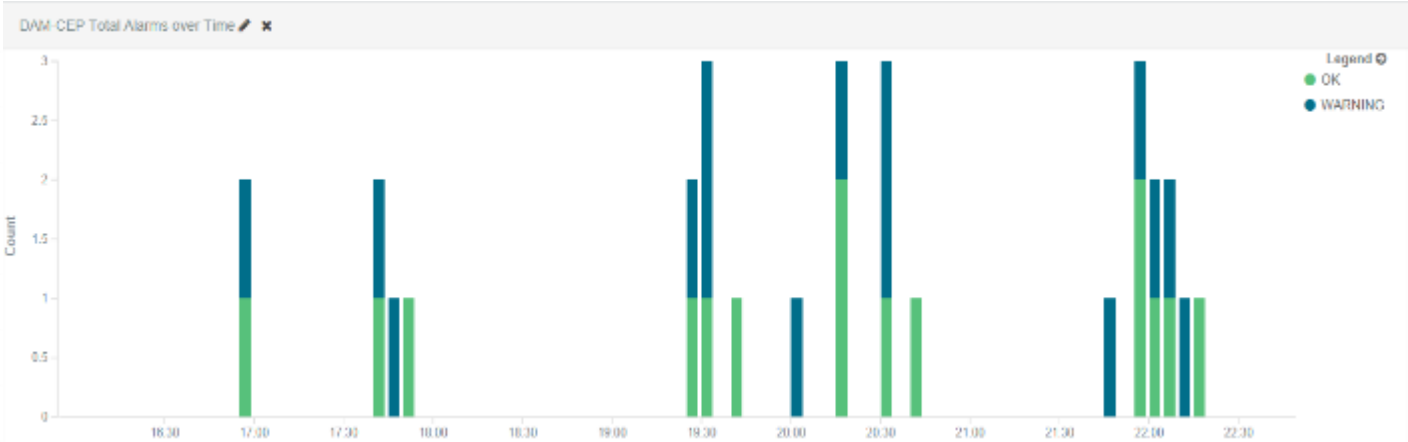
www.cern.ch

Challenges in alarming

host flapping and spamming



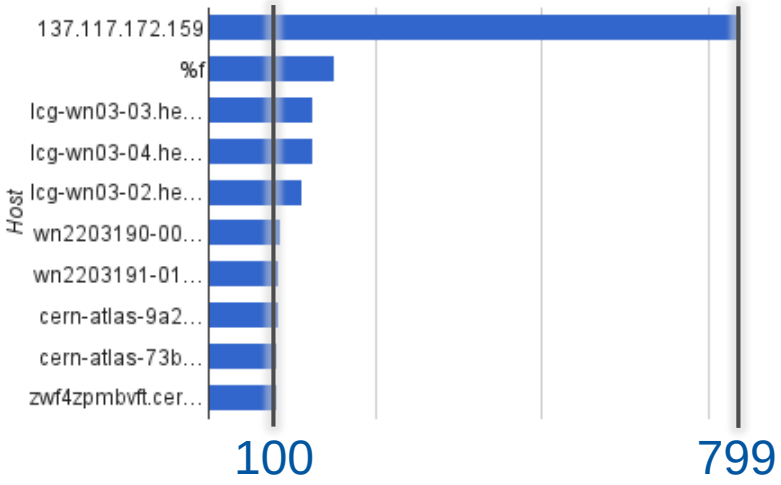




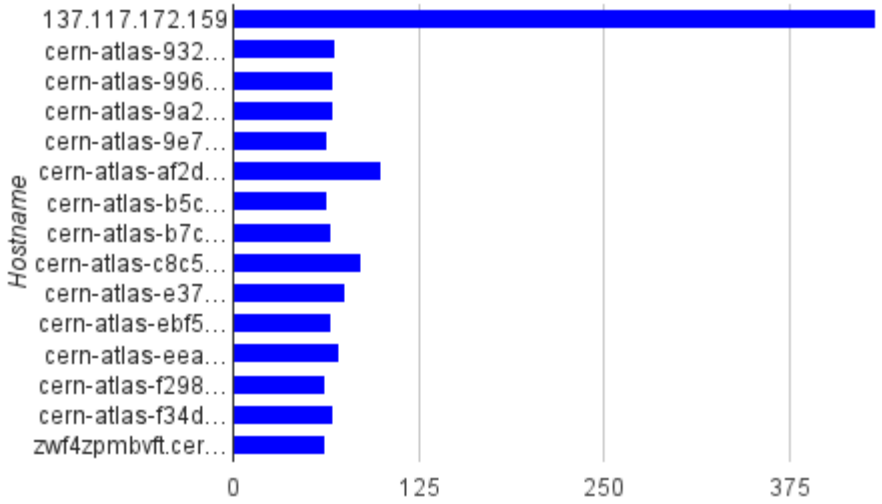
First statistics from live deployment

analyzing output of first two runs connected to production servers

Notifications per host in first run



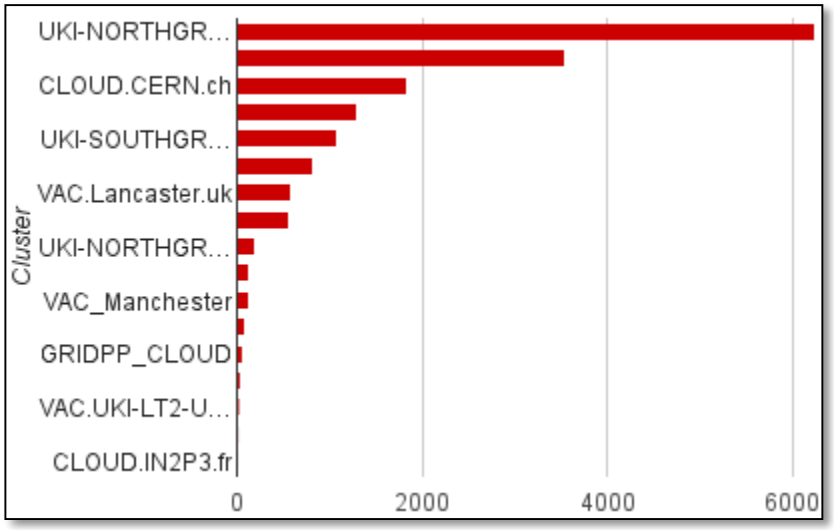
Notifications per host in second run



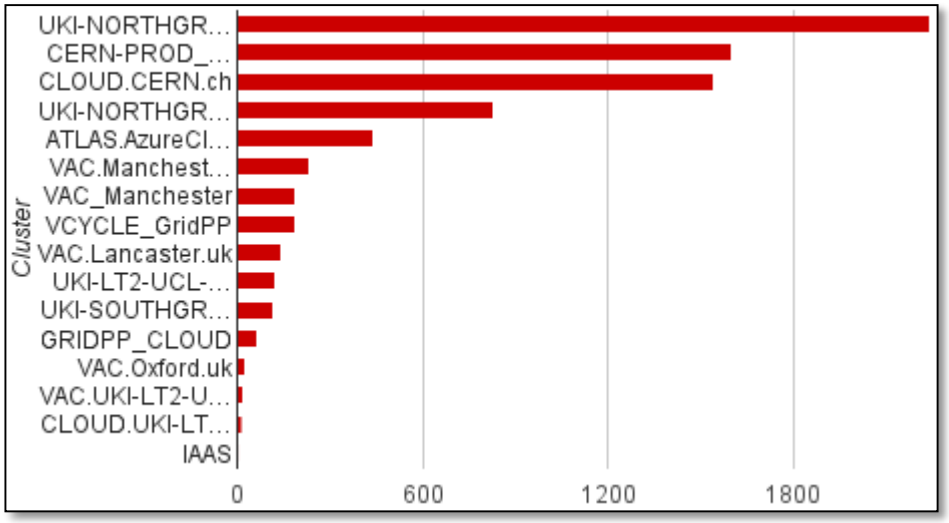
First statistics from live deployment

analyzing output of first two runs connected to production servers

Notifications per cluster in first run



Notifications per cluster in second run



First statistics from live deployment

analyzing output of first two runs connected to production servers