Advanced Monitoring With Complex Stream Processing

IT Technical Forum

Magnoni Luca, Cons Lionel IT-SDC
Reguero Ignacio IT-PES
Few words about myself

• Software Engineer
• Experience with monitoring projects
• Worked@PH/ATLAS-TDAQ
  • Automated/Reactive monitoring at speed
• Work@IT/SDC-MI
  • Monitoring WLCG with lambda-architecture
  • Advanced Monitoring for Messaging services
Outline

• Messaging service and monitoring
• Technology investigation
• Building a solution
• Summary
Messaging service @ CERN

Run Services

Provide Additional Tools

Offer Advanced Monitoring

Producer

Messaging service

Consumer

Consumer
Messaging infrastructure

Host
Broker
Broker
Broker

Host
Broker
Broker
Client

Host
Broker
Broker Cluster
Client

16 Physical, 9 Virtual Machines
23 Brokers
10 Clusters
157 Clients
up to 150 Millions msg/day
Monitoring: measure everything

Host

broker.cpu
broker.heap
broker.connections
broker.received_msg
broker.sent_msg
...

Metric

broker.stomp
broker.http

Check

client.received_msg
client.sent_msg
client.stored_msg

Log

host.syslog
host.iptables
host.SELinux

Check

broker.httpd
broker.activemq

Check

host.cpu
host.memory
host.swap
...

Check

Client

Client

broker.stomp
broker.http

Client
We have a situation… or not?

Number of messages not consumed (stored) for a specific client on different brokers

YES
Better call someone!
We have a situation… or not?

Number of messages not consumed (stored) for a specific client on different brokers

NO
Everything is fine.
We have a situation… or not?

Number of messages received from a specific client on different brokers

NO
Everything is fine.
We have a situation… or not?

Number of messages received from a specific client on different brokers

YES
We have a situation... or not?

Number of network connections on different brokers

YES
Unbalanced use of brokers in a cluster.
What we need

- **Detect problems** from metrics, logs and checks
- **Notify users** when something is *not ok* (but also when it *goes back to ok*)
- **Automate** as much as possible
Monitoring is complicated

- **Information is scattered**: value not in the single event but in the overall system behavior over time
- Events have to be **correlated and aggregated**
- Fault diagnosis requires **competence and experience**
Technology investigation
Quest for technology

- Stream Processing frameworks
  - Storm, Spark Streaming, Akka, Samza, etc.
  - emphasis on streaming high data volumes
  - basic analytic primitives, derived from batch processing
- Quite surprisingly, not much from the monitoring world
- Process streams of information at high rate with complex time-based analysis is not new
  - financial analysis, sensor networks, business intelligence,…
Complex Event Processing (CEP)

- A.k.a Event Stream Processing (ESP) / SQL streaming analytics
- A set of technology to **process and discover complex patterns among streams of events**
- A cross between DBMS and Rule Engines
  - formalized by Prof. D. Luckham (Stanford) in 2002
  - *specialized system for special purpose* (Prof. Stonebraker/StreamBase)
- Several providers (commercial & open source)
  - Oracle, IBM, TIBCO, Drools Fusion, Siddhi, EsperTech
- Functional properties
  - stream analysis with Domain Specific Language (SQL-like or custom)
CEP: continuous-processing model

Move from a traditional **pull-based store-and-analyze** model
data stores grows permanently, search becomes expensive

To a **data-driven, in-memory, on-the-fly** analysis
continuous query evaluation model
CEP: a database upside/down

Store the data. Compute on demand.

SQL

select count(*)
from circle
group by color;

~ SQL

select count(*)
from circle.win:time(1 hour)
group by color;

Store the query. Continuous computation.
Esper: open source CEP/ESP engine

Esper [...] provide a highly scalable, memory-efficient, in-memory computing, SQL-standard, minimal latency, real-time streaming-capable Big Data processing engine for historical data, or medium to high-velocity data and high-variety data.

- **GPL**, with commercial support and extensions
- strong community, support and documentation
- long list of big customers (new entry: Ebay with Pulsar)
- in use at CERN:
  - **PH/ATLAS TDAQ**: Intelligent monitoring and real-time analytics for ATLAS-TDAQ: a CEP-based solution
  - **BE/Control**: Surveillance of CERN Accelerator Logging Processes with CEP and Esper
Esper

- **Lightweight Library**
  - Java, standalone or embeddable
  - input-output connectors: Java API, HTTP, JMS, AMQP, DB, File
  - data format: JSON, XML, custom
  - supports custom adapter
  - strong performance
    (500000 events/s with 3\(\mu\)s avg latency)

- **Feature rich** SQL-like Event Processing Language (EPL)
Esper analytic features

Data windows
to instruct the engine on how long to retain events. (Time or length based, with sliding or batch behavior)

Patterns
for pattern matching based on logical and temporal event correlation, or regular expression. (Match-recognize, timer-control patterns)

Joining data
across stream or with external sources, for easy integration with external data sources such as web services and plain DB.

Event analysis
grouping, aggregation, rollup, cube, sorting, filtering, transforming, merging, splitting or duplicating of event series or streams.

Pluggable functions
for event pattern and event stream analysis via user-defined functions, plug-in views, plug-in aggregation functions. (Java callback, Script support, Lambda functions)

Time Control
supports externally-provided time as well as current system time, allowing applications full control over the concept of time within an engine and full control over which thread(s) evaluate timer schedule for queries.

Joining data
across stream or with external sources, for easy integration with external data sources such as web services and plain DB.
EPL: event selection

- How do I look for specific events on a stream, dropping the unwanted events?

  \[
  \text{select \ } * \ \text{from \ SensorEvent \ where \ temperature \ > \ 90}
  \]

- How do I aggregate several (simple) events from a stream into a single new (complex) event summarizing event properties of the simple events?

  \[
  \text{select \ \text{avg}(temperature) \ from \ SensorEvent}
  \]
EPL: data windows and patterns

• How can I detect N events within X seconds?

```sql
select * from MyEvent(type=ERROR).win:time(3 min)
having count(*) >= 5
```

• How do I use patterns to correlate events?

```sql
select * from pattern [every s=StartEvent ->
a=EndEvent(id = s.id)]
```

• More examples from Esper’s “Solution Patterns”
EPL programming model: pipes and operators

Input channel → EPL statements → Streams → Output channel

Streams: Input channel → EPL statements → Output channel
Building a solution
Messaging Monitoring model

* Deeply inspired by the Nagios model
Messaging Monitoring architecture
Metis

- Esper-based service
- Java & Spring
- Configurable Input/Output channels to receive and to report events
- Digests JSON-structured monitoring data
- Implements the messaging monitoring model
- EPL statements as monitoring directives
- Run-time admin interface
Detect a problem from a Metric
Check generation: threshold

• Check = f (Metric)

```sql
// Check load per host
//
insert into Check
select host as m.host,
case when m.value > 95
    then {STATE.CRITICAL, 'Load is too high: ' || m.value}
when m.value > 90
    then {STATE.WARNING, 'Load is high: ' || m.value}
when m.value >= 0
    then {STATE.OK, 'Load is ok'}
end as result
from Metric(name='load') as m;
```
Detect a problem from multiple Metrics over time
Metric transformation: time-aggregation and grouping

- Metric = f (Metric)

```sql
//
// Avg stored msg per broker over sliding 1 hour
//

insert into Metric
select
  m.host as host,
  m.broker as broker,
  'stored_avg' as name,
  avg(m.value) as value
from Metric(grp='msgbrk', name='stored')
  .win:time(1 hour) as m
group by host, broker;
```
Metric transformation: counter to rate

Plot shows client messages received (counter) by broker
Metric transformation: counter to rate

Plot shows client messages received (rate) by broker.
Counter to rate operator

- Metric = f (Metric_x, Metric_{x-1})

```javascript
// EPL Lambda function to compute rate
create expression computeRate {
    m =>
        (m.value - prev(1, m.value)) / (m.timestamp - prev(1, m.timestamp))
};
```
Hysteresis operator (e.g. WARNING >90 and OK <60)

- **Check = f (Metric, Check)**

```cpp
//
// Script to compute Hysteresis thresholds
//
create expression double mvel:hyst(target_state, previous_state, up_threshold, down_threshold) [
    if (target_state > previous_state)
        return up_threshold;
    else
        return down_threshold;
]
```
Log as first-class citizen

- Check = f (Log)
- Additional feature: log parsing
- Log = f (Log)
  - Grok library from Logstash
  - Re-use Logstash patterns

```java
//
// Log parsing using via
// the Grok library
//
insert into StructuredLog
select MetisUtils.GrokParse(log.text)
from Log(name=“syslog”) as log;
```
Status

- Status = f (Check, Status) // Check Validity.
  // CHECK goes to UNKNOWN after 30 // minutes without new check event
  //

insert into Check
select
  check.*, STATE.UNKNOWN as value,
  "No new check since a while..." as text
from
  pattern [ c = Check -> (timer:interval(30 min) and not c_up=Check(FID=c.FID))];
Metis: project status

- **In production** since December 2014
- **Data types/instances:**
  - **Metrics** 97 types, 6179 instances
  - **Logs** 6 formats, 276 instances
  - **Checks** 16 types, 673 instances
- **Data-rate:**
  - 100 metrics/second
  - 10 logs/second
  - 13 checks/second
Metis: features summary

- Metric transformation
  - grouping and aggregation across time and brokers
  - counter/rate
- Check generation
  - threshold
  - hysteresis
  - patterns
- Status inference
- Notification logic
  - when to report to users and what
- EPL templates as monitoring directives
  - fine grain configuration
Does it work?

Yes 😊

From: CERN Messaging Team <mignr@cern.ch>
To: <rucio-alarms@cern.ch>
CC: CERN Messaging Team <mig@cern.ch>
Subject: CRITICAL: too many rucio-tracer-kronos messages stored on the atlas brokers
Date: Thu, 28 May 2015 06:22:12 +0200

There are too many (100507) messages stored on the atlas brokers for your client rucio-tracer-kronos (/queue/Consumer.kronos.rucio.tracer), see: https://mig-graphite.cern.ch/grafana/#/dashboard/file/client.json?var-client=rucio-tracer*

Please make sure that your consumers are working correctly.

Regards,

CERN Messaging Team - mig@cern.ch - http://cern.ch/mig
Does it work?

Yes 😊

From: CERN Messaging Team <mignr@cern.ch>
To: <rucio-alarms@cern.ch>
CC: CERN Messaging Team <mig@cern.ch>
Subject: OK: rucio-tracer-kronos messages stored on the atlas brokers
Date: Thu, 28 May 2015 06:32:12 +0200

FYI, the number of rucio-tracer-kronos messages stored on the atlas brokers is now back to normal.

Regards,

CERN Messaging Team - mig@cern.ch - http://cern.ch/mig
Summary
Lessons learned

• **CEP**: a powerful and versatile technology
  
  • consider CEP when *aggregation and correlation over time is a key feature* in your analysis

• **EPL/SQL-like language** makes complex analysis easier
Lessons learned

• Esper is good at what it does
  • but needs several things to build a service
• Spring is good to glue all these things together
• Esper has a quite steep learning curve
  • but this pays off in the end
• Esper features allow for very advanced monitoring rules
  • getting close to intelligent monitoring
• A flexible transport layer eases integration and testing
Lessons learned: test everything

- Unit testing is crucial
- Esper allows to control time (wow!)
- Replay old events
- Mock time-based sequence
- Run development Metis (e.g. from git checkout) on the real data flow on the preferred IDE…
Consideration on scalability

- Metis runs on a modest VM (4 cores/8GB)
- ~ 80 MB Heap, ~ 6% CPU
Consideration on scalability

- **Dimension your system to make analysis, not to store data**
- **Vertical**
  - EPL processing optimized (decision tree) with fine grain control on multi-threading
- **Horizontal**
  - Multiple instances (stream partitioning)
    - Ebay/Pulsar architecture (with Kafka and Zookeeper)
    - Esper within streaming framework:
      - Esper/Storm and Esper/Akka are working example
Roadmap

- GNI/ServiceNow integration
- Roger integration
- State preservation
- Shadow instance
- Do more with logs
Summary

• We have an advanced monitoring system
  • used by us and by our clients
• Esper has proven to be a good choice
• The same approach could be useful to other services
  • Metis provides a generic monitoring model with a modular architecture
Thank you! Questions?