Real-time statistic analytics for the WLCG Transfers Dashboard with Esper

Project Report - CERN Summer Student Program 2014

IT-SDC-MI

Author: Maria Varvara Georgiou *Supervisor:* Luca Magnoni

Acknowledgments

I would like to thank my supervisor Luca Magnoni for his continued support and careful guidance, as well as the rest of the IT-SDC-MI section for their help over the course of this project.

Contents

1

Introduction

1.1 WLCG Data Transfer Dashboard

The WLCG Data Transfer Dashboard[1] is monitoring the data movement, which are generated by the LHC experiments, between sites and scientists around the world. Currently all the transfer information data are recorded inside an Oracle database and the statistics are computed on a regular basis using PL/SQL procedures.
This is a solid and reliable solution, but it can be improved because the PL/SQL procedures are not scaling with the increasing data volume and also the statistics re-computation may take days/weeks.
This project aims to integrate Esper, an open-source in-memory processing engine, to the existing work flow to allow a real-time computation and visualization on fresh data and to speed-up all the statistics generation. This project is part of the WLCG monitoring analytics framework[2] evolution.

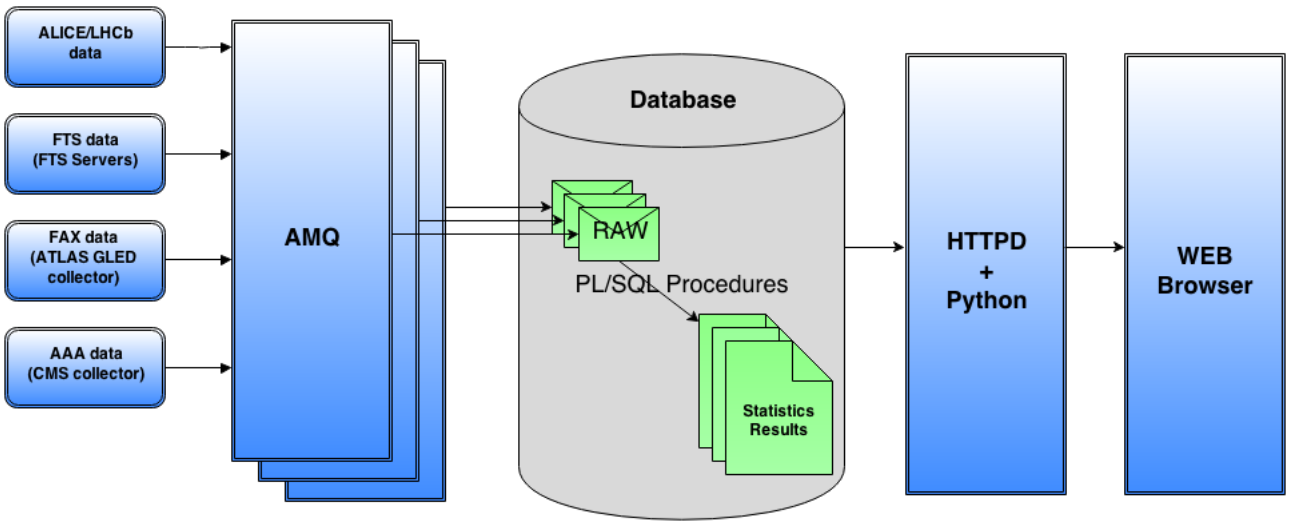

Figure 1.1: An overview of the current system architecture

2

CEP/Esper

The need for fast in-memory computation is not new. It is needed in several fields like financial analysis, wireless networks etc. Complex event processing (CEP) technologies were created in order to serve this need by processing streams of events at high rate with low latency. The most widely adopted open source engine for this purpose is Esper, which is the main tool used for the development of this project.

2.1 What is Esper?

The need for fast in memory computations is not something new. Complex Event Processing (CEP) technologies were created to process streams of events at high rate with low latency. The most widely adopted open source/GPL is Esper. It enables rapid development of applications that process large volumes of incoming messages or events, regardless of whether incoming messages are historical or real-time in nature.

2.2 Event Processing with Esper

Esper analyses data with EPL (Event Processing Language) which is SQL-like. In contradiction to the relational database model Esper stores the query, which is continuously running. It also stores the results if necessary but not the data.

$$SQL \longrightarrow EPL$$

$$Rows \longrightarrow Events$$

$$Tables \longrightarrow Event\ Streams$$

Figure 2.1: a mapping between Esper and the relational database model

3

The project

Tasks

The project consisted of the above tasks:

- Get familiar with Maven, JUnit, Git and Esper
- Understand and document the existing PL/SQL procedures
- Implement the necessary modules
- Implement the procedures with EPL
- Unit test all the modules

3.1 Procedures

There are six procedures that are applied periodically on the recorded data in order to extract statistics. Those procedures are described in the twiki page of the WLCG monitoring analytics framework project[2]. All of them are applying different kind of aggregations on the data in order to summarize the data transfers between two sites according to time-unit, type, of operation (read or right), etc.

Those procedures were:

- Ten minutes aggregation
- Ten minutes aggregation for CERN data storage
- One hour aggregation
- One day aggregation
- Access Pattern Statistics
- User Statistics

3.2 Project Architecture

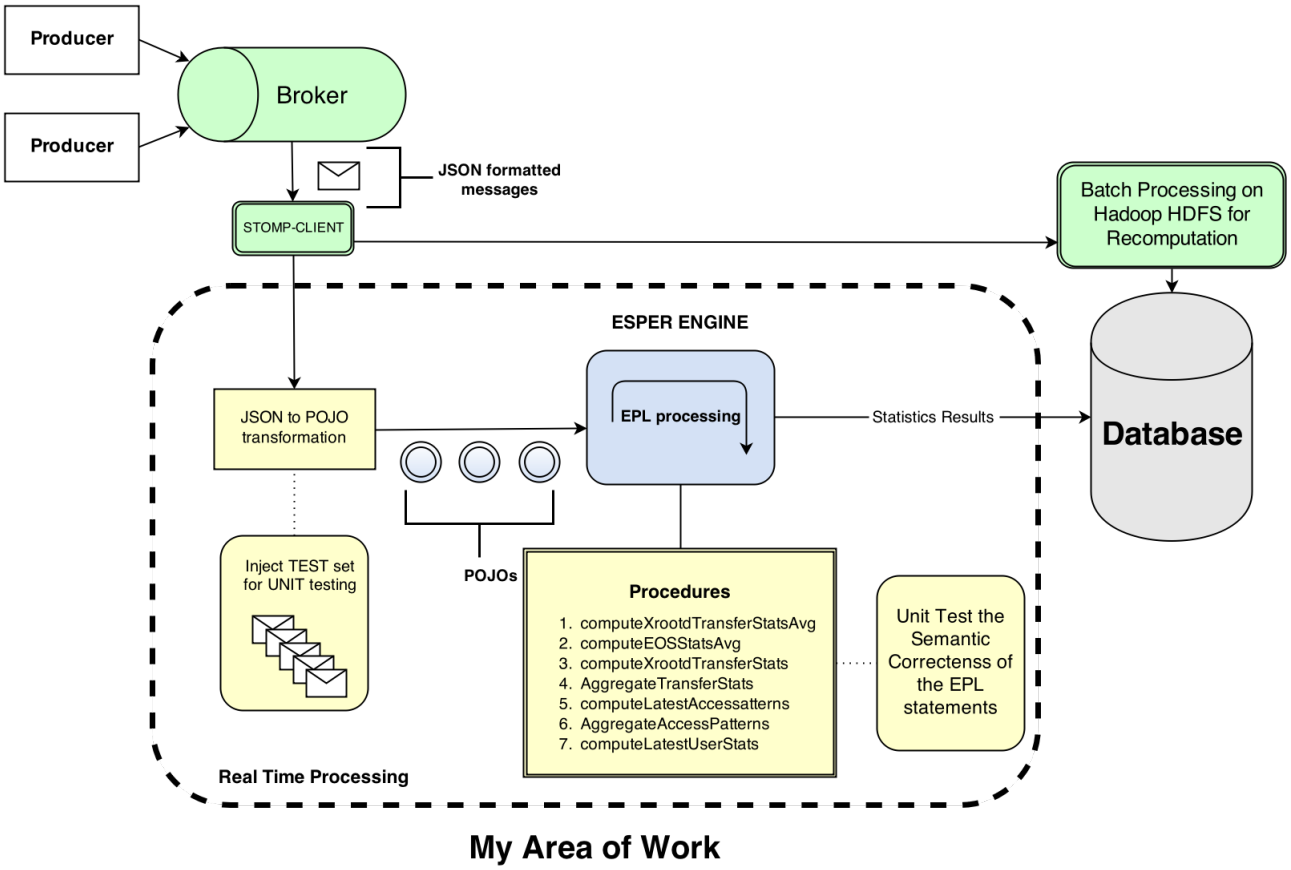

Figure 3.1: An overview of the system architecture with the Epser module

3.2.1 JSON To POJO transformation

The data movement between different sites around the world is recorded into log messages. These log messages store information such as the transaction start and end time, the source and the destination site,the kind of the transaction (read or write) etc.

Log messages are distributed by the message broker and initially are JSON form. The data in that form are not ready to process with Esper, so we need to perform a few preprocessing steps before we inject them to the Esper engine. First step is to subtract parts of the message that are not needed and second to transform the JSON file into Java object (POJO). Due to the amount of properties (48) that the object consists of the Builder pattern was used in order to be easier to handle it.

3.2.2 EPL Processing

This is the module where the implemented EPL statements are continuously running. A listener is invoked periodically in order to check for incoming events and process them according to the statement.
The implemented EPL procedures follow a map-reduce approach.

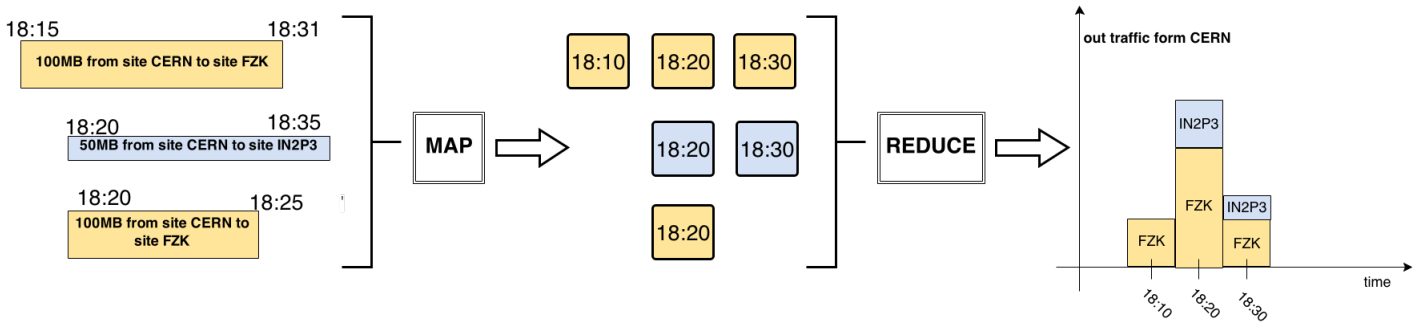

Figure 3.2: An example of the map reduce approach on EPL statements implementation

Figure 3.2 shows an example of the map reduce implementation for the 10 minutes aggregation. Three different events, which represent transfers from the CERN site to two other sites (FZK and IN2P3 respectively), are injected in the map statement. The map statement is splitting the incoming events into smaller pieces according to the time bins that they belong. E.g the first event belongs to the time bins 18:10, 18:20, 18:30 thus it is split in three different events which afterwards are injected into the reduce statement which finally aggregates them into the final results.

3.2.3 Example of EPL statement: 10 minutes aggregation

1. `Inject Log Message event into MAP statement`
2. `Split the Log Message into several Log Map Events according to the time bins the initial event belongs`
3. `Inject each of the Log Map Events into a Single Log Statistic Event and compute the following:`
   (a) `If writes_bytes_at_close>0 then we have a client_domain else we have a server_domain and set it as src_domain`
   (b) `If read_bytes_at_close>0 then we have a server_domain else we have a client_doma and set it as dst_domain`
   (c) `if client_domain=server_domain set remote_access 0 else set is as 1`
   (d) `if writes_bytes_at_close+read_bytes_at_close=file_size set is_transfer=1 else is_transfer=0.`
   (e) `if read_bytes_at_close>0 then setactivity='r'`
   (f) `if write_bytes_at_close>0 then set activity='w'`
   (g) `if write_bytes_at_close<=0 and read_bytes_at_close=<0 then set activity='u'`
4. `Aggregate all the single log statistics`

(a) `If there is not already a time bin for the injected Single log statistic event then create it and insert:`
   i. srcDomain
   ii. dstDOmain
   iii. isRemoteAccess
   iv. usrProtocol
   v. isTransfer
   vi. Activity
   vii. periodEndTime
   viii. active
   ix. bytes
   x. activeTime
   xi. updateTime

(b) `Else update the existing bin:`
   i. active=active+newSingleLogStatistic.active
   ii. bytes=bytes+newSingleLogStatistic.bytes
   iii. activeTime=activeTime+newSingleLogStatistic.bytes

3.2.4 Unit Test & Time Control

In order to test the EPL statements' semantic correctness unit tests were performed. A data set of test messages was created and injected to the Esper engine. Due to the fact that the statements are working with specific time windows, those had to be simulated as well. Esper provides the ability to use externally-controlled time, giving your application full control over the concept of time within an engine or isolated service.
The rest of the project components were tested as well.

4

Results and Future Work

4.1 Results

First conclusion is that it is possible to compute the dashboard transfer statistics with Esper.The EPL statements that are so far implemented are the ten minute, one hour and one day aggregation.
The advantages of using Esper are many. One can do incremental updates than full re computation. In this way the results can be provided as the new data arrive without delay and allow fresh data visualization.

4.2 Future Work

- One limitation of the data collector system which should be improved in the future is that despite the effectiveness of the computation, the system is bound to the fact that the logs are received only after the transfer ends
- Esper by design is a single host processing engine, something that raises considerations over the scalability and availability of this solution so an interesting work could be a real measurement of the log messages that can be injected in the engine to the corresponding memory footprint. Nevertheless Esper is designed to process thousands of events per second and in this specific use case we are in the level of tens messages per second.
- Also when the system will be improved with partial transfer log updates (log messages for an active transfer), there will be no need to change the EPL procedures.

References

Joshua Bloch, I. (2008). *Effective Java*, Addison Wesley; United States

[1] Dashboard project:
http://dashb-wlcg-transfers.cern.ch/ui/#,
http://dashb-atlas-xrootd-transfers.cern.ch/ui/#,
http://dashb-cms-xrootd-transfers.cern.ch/ui/#

[2] WLCG monitoring analytics framework:
https://twiki.cern.ch/twiki/bin/view/LCG/WLCGMonDataAnalytics

Esper: http://esper.codehaus.org/

git repository: http://git-scm.com/book/en/Git-Basics-Getting-a-Git-Repository