Enabling Schema Agnostic Learning Analytics in a Service-Oriented MOOC Platform

Jan Renz
Hasso Plattner Institute
University Potsdam
jan.renz@hpi.de

Gerado Navarro-Suarez
Hasso Plattner Institute
University Potsdam

Rowshan Sathi
Hasso Plattner Institute
University Potsdam
rowshan.sathi@hpi.de

Thomas Staubitz
Hasso Plattner Institute
University Potsdam
thomas.staubitz@hpi.de

Christoph Meinel
Hasso Plattner Institute
University Potsdam
christoph.meinel@hpi.de

L@S 2016, April 25-26, 2016, Edinburgh, Scotland UK
ACM 978-1-4503-3726-7/16/04.
http://dx.doi.org/10.1145/2876034.2893389

Abstract
This paper at hand describes the design and implementation of an analytics service to retrieve live usage data from students enrolled in a service-oriented MOOC platform for the purpose of learning analytics (LA) research. A real-time and extensible architecture for consolidating and processing data in versatile analytics stores is introduced.

Author Keywords
Service oriented architecture, MOOC, Learning Analytics

ACM Classification Keywords
H.4 Information Systems Applications; H.5 Information interfaces and presentation; K.3.1 Computer Uses in Education;

Introduction
To be able to cope with the "massiveness" of MOOCs, the underlying Learning Management Systems (LMS) must be able to serve thousands of users at the same time. Service Orientated Software Architecture (SOA) using Microservices [1] is a popular pattern to achieve the required performance and scalability.

Figure 2: Experience Graph Schema showing the interconnection among different entities such as users, courses and learning items

Figure 3: Use Case Example: Showing the geo position of users To implement this feature and additional *transform* step has been added that converts the Users IP-Address to an approximate Geo Information and adds this information to the the tracked event. This information is then queried using *ElasticSearch*.

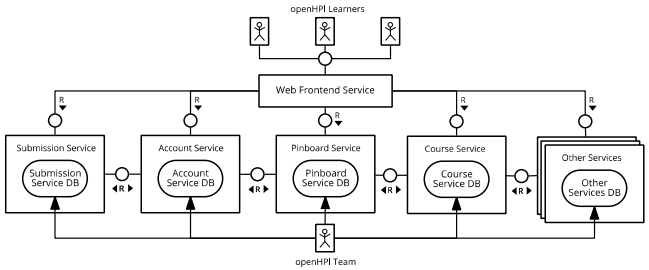

Figure 1: FMC diagram of an service orientated LMS architecture. While this architecture uses stand alone components that communicate with each other, it is notable that this is an internal architecture only.

Given a data analysts view, one of the major drawbacks of such a distributed architecture is the loss of a centralized place, where all the data (including all the learning analytics data) is stored. This complicates the retrieval of data, especially if the required data is spread across multiple services. Given an architecture as proposed in [2] and shown in Figure 1, a learner taking a course will produce data in the data storages of several services.

This paper will discuss two main research questions:

1. Facing a service-oriented MOOC platform, what is the best approach to collect and extract the relevant data?
2. What are appropriate schemas and technologies for storing, analyzing, and querying the data?

Service specific and cross service data

All data, that is contained within a single service can be retrieved by just querying this specific service. Given the course enrollments are managed by a specific service, retrieving data like enrollments per course or enrollments over time can be retrieved by querying this service or the underlying database. This does not require a dedicated analytics service.

But as soon as the required data is spread across multiple services (for example the overall course activity) or the data does not belong to any specific service (such as, e.g. UI interaction tracking data) a new service must be introduced to take responsibility of these tasks. We will call this service the Learning Analytics Service (Lanalytics). Its responsibility is the data collection, transformation, storage, and retrieval of all required learning analytics data. The service concept in brief is, an analytics service which collects or gathers the latest data from different services, perform additional processing steps, transforms the gathered data into different data schema and finally loads the transformed data into several data stores, called Analytics Stores (AS). Researchers and other services can access these AS to retrieve data or perform analyses.

Data collection

Traditional approaches, such as ETL (Extract, Transform and Load process), do not work in the current architecture as it does not support real time data processing and short term load peaks, which violates SOA principles, as it collects data directly from databases. Pulling data from the service interface also does not meet the data collection requirements of service oriented architecture. Data virtualization is an emerging approach which provides an abstraction layer on top of the database. This abstraction layer provides a unified access to retrieve data from the underlying infrastructure. The drawback of this approach is its complexity and the lack

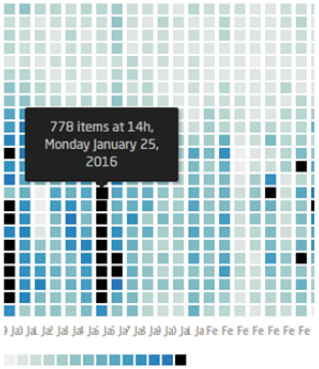

Figure 5: Use Case Example: Showing the overall course activity by hour.

of complete transparency for data and information management [3].

Therefore, an event driven push model for services is an appropriate option for data collection in this context. A combination of SOA and event-driven architecture is proposed as a solution for data collection, which is known as 'Event-Driven SOA' and used for this service.

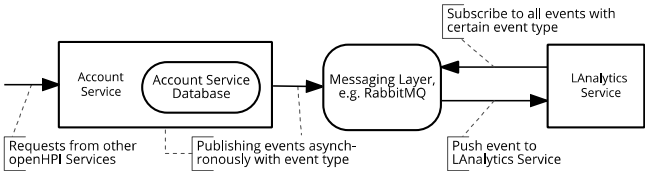

Figure 4: Overview of the event-driven concept for data collection

The events that are sent by the services follow a format inspired by the Experience API [5] in the form of: <Actor> <Verb> <Object>, with <Result> in <Context>. As shown in Figure 2 the *Lanalytics* service then subscribes to this events.

Data transformation and storage

After the service has received the subscribed events this data has than to be processed and stored. The service currently supports three different data schema, which will be described in the following paragraphs.

The **Experience Event schema** stores the processed events in *ElasticSearch* and *Postgres* as flat event data. The **Experience Graph Schema** uses the same data, but represents it in a graph model**.** Apart from directly mapped entities, there are some entities which describe the relationship among different nodes and thus, they are explicitly mapped to the edges. In the graph this relationship is seen as a statement where *Actor* is the origin of the edge, *Verb* is the type of the edge and *Object* is the destination of the edge. Additional parts of the statement such as *Result* and *Context* are added as properties of the edge. Therefore, by following the mentioned rules it is possible to create a graph that contains a) most entities of the platform, b) the relationships between these entities, c) the user activities represented similar to the Experience API statements, and d) the student profile data as the students are represented as nodes with all available data attributes. Figure 2 not only shows the interlinked data model but also shows the possibilities of finding similarities among learners and to compare the learning experience and outcome between two learners. This data is stored in a Graph database. Furthermore, it supports **the MOOCdb schema[8]**.

Data transformation is based on a processing pipeline, inspired by the *chain responsibility* and *command* pattern [6]. Each pipeline consists of *ExtractSteps*, *TransformSteps* and *LoadSteps* classes. *ExtractSteps* processes the raw input data into a container class *ProcessingUnit*. Then the pipeline passes the *PipelineUnit* to *TransformSteps,* which transforms the data by implementing a *transform()* method. The mapping is done in the *TransformStep*. As a part of this mapping, *TransformSteps* creates *LoadSteps*. A layer *LoadCommands* is introduced between *TransformSteps* and *LoadSteps* to ensure the modularity and extensibity of the LAnalytics Service. Finally, The *LoadSteps* excecutes the *LoadCommands* to update the respective analytics stores. The stored data can be retrieved via a REST interface. An example how this could be done can be found in [7].

Performance and scalability evaluation

As this communication is asynchronous, data processing will not not slow down the overall performance of the system on load peaks. If more events are sent than the Lanalytics service is able to process in real time, it will result in an increasing cue size and receiving slightly outdated data when retrieving data from this service. Given the service based architecture and the decoupling from other system components it can easily be scaled by deploying multiple instance or scaling the underlying data storages.

Conclusion

Introducing a dedicated service enables learning analytics for data that is spread across multiple service or not maintained by any service at all (, such as UI interactions). By using multiple analytic stores with different underlying technologies the advantages of these stores can be utilized to do complex analytic queries at a low implementation cost. While some storages and schemas are good for event based data, other represent graphs of interconnected entities. Since the rollout of the first version, most analytics tasks have been implemented using the the *Experience Event* schema running on ElasticSearch.

The major drawback is that every storage system has to be maintained. Furthermore, computing power and storage has to be provided for every single storage system.

References

1. Sam Newman. 2015. *Building Microservices*. " O'Reilly Media, Inc.".
2. Christoph Meinel, Michael Totschnig, and Christian Willems. 2013. openHPI: Evolution of a MOOC platform from LMS to SOA. In *Proceedings of the 5th International Conference on Computer Supported Education (CSEDU), INSTICC, Aachen, Germany*, Vol. 5.
3. Arcot Rajasekar, Mike Wan, Reagan Moore, and Wayne Schroeder. 2006. A prototype rule-based distributed data management system. In HPDC workshop on Next Generation Distributed Data Management, Vol. 102. Citeseer.
4. Learning Measurement for Analytics Whitepaper, https://www.imsglobal.org/sites/default/files/caliper/IMSLearningAnalyticsWP.pdf
5. Advanced Distributed Learning. 2013. The Experience API - Version 1.0.1. (2013). http://www.adlnet.gov/wp- content/uploads/2013/10/xAPI_v1.0.1- 2013- 10-01.pdf
6. Rebecca Ferguson and Simon Buckingham Shum. 2012. Social learning analytics: five approaches. In *Proceedings of the 2nd international conference on learning analytics and knowledge*. ACM, 23–33.
7. Jan Renz, Daniel Hoffmann, Thomas Staubitz, and Christoph Meinel, 2016, Using A/B Testing in MOOC Environments in *Proceedings of the 6th International Conference on Learning Analytics and Knowledge, ACM, http://dx.doi.org/10.1145/2883851.2883876*
8. Kalyan Veeramachaneni, Sherif Halawa, Franck Dernoncourt, Una-May O'Reilly, Colin Taylor, Chuong Do. MOOCdb: Developing Standards and Systems to Support MOOC Data Science. arXiv# 1406.2015