Towards a Semantic Definition for a Cloud based Event Notification Service

Marc Schaaf[1], Stella Gatziu Grivas[1] and Arne Koschel[2]
[1]*University of Applied Sciences Northwestern Switzerland, Riggenbachstr. 16, Olten, Switzerland*
[2]*University of Applied Sciences and Arts, Ricklinger Stadtweg 120, Hannover, Germany*

Keywords: Event Processing, Cloud Computing, Portability.

Abstract: We propose an Activity Service as a component in cloud computing. As a particular novelty we base this Activity Service on the well-defined and proven semantics of Active Database Management Systems (ADBMS) to overcome the challenges of developing portable cloud based event processing applications. As contribution of this paper we provide the first steps of the adaption of the ADBMS semantics to the distributed and highly dynamic environment of the cloud.

1 INTRODUCTION

Cloud Computing is a new paradigm, it offers vast resources and highly dynamic scalability while reducing the required upfront investment cost to a minimum. However the development of cloud enabled applications considering benefits like the increased agility resulting from the possibility to dynamically utilize resources from a vast resource pool is still a hard task. Suitable software architectures accompanied by the appropriate cloud services are required to develop cloud enabled applications.

We are convinced that the combination of the cloud paradigm with event processing provides a promising architectural approach for designing applications able to benefit from the cloud's flexibility. We see this as a natural extension of the well established active mechanisms in database systems that over time found their way into the world of distributed systems in many incarnations. Furthermore for many "new" application domains (like smart applications, emergency management), event driven approaches offer a capable foundation for coping with challenges resulting from the constantly increasing amount of data and near real-time analysis requirements. However event processing services usable for cloud applications with well defined semantics which provide a reliable foundation to build scalable *and* portable applications are still missing. Many cloud offerings include services that allow building event processing applications like for example Amazons Simple Queuing Service. However these services are highly provider specific and don't feature a well defined and proven semantic. Current standardization approaches like for example OpenStack are mostly focused on infrastructure aspects and don't cover any standardized support for event processing based applications.

A simple example, which illustrates the need for detailed semantic definitions, is the event parameter time. A time like "04:37:21" is problematic when considered that time indications depend on the location where they are made. Thus, a precise time indication needs additional information, such as 'according to UTC'. The time is one simple example required for an event itself. But many other questions are also unanswered as they arise from the used notification service like for example the handling of out of order or missing events or the behavior of the service implementation when the processing of an event fails.

As illustrated, the semantics of an event payload and the processing conditions are not explicitly described and require further knowledge to allow the processing without knowledge of the event producers and the notification services semantics. Current cloud based services that can be used to build event processing systems lack the capability to explicitly specify such aspects. As a result, when building an application based on these services, the application becomes highly depended on the implicit, provider specific semantics, which results in a high risk of a vendor lock-in. This even in addition to the risks introduced by using a provider specific API. We see this high risk of a vendor lock-in as one of the major challenges when developing cloud based event applications.

To overcome this issue we propose the Activity Service as a provider independent service with a clearly defined semantic for building event driven cloud applications. The contribution of this paper to the Activity Service lies in the discussion of the initial steps of its semantic definition. Discussions regarding the architecture and first realizations have been presented in (Schaaf et al., 2010), (Schaaf et al., 2011).

A few other approaches for cloud based event processing exist like (Goyal et al., 2009). At least some combination of event driven and service oriented architecture for the cloud is discussed there but remains quite high-level and in particular focuses on policy-driven event processing for the cloud. Some event monitoring and propagation within the cloud in conjunction with complex event processing (CEP) is discussed in (Wishnie et al., 2009). However they do not really address an Activity Service for the cloud at all and in particular not with well-defined semantics.

The remainder of this paper outlines our approach followed by a quick overview of the Activity Service architecture. Afterwards we discuss the required adaption of the initial ADBMS semantic parameters and continue with the discussion of the event semantic parameters. We conclude with a brief overview of our implementation, a summary of our results and an outlook of our next steps.

2 OUR APPROACH

To overcome the mentioned challenges, we define the Activity Service as a general event notification service for cloud environments that features a well defined semantic. With the availability of the Activity Service we will help cloud application developers to overcome the afore mentioned challenges when developing scalable and portable cloud applications.

Our approach features a specific focus on the semantics of the processing system as a comprehensive definition is currently missing for distributed event based systems. The semantic foundation of our work is based on well-established earlier work from ADBMS (Paton, 1999), (Widom et al., 1996). In particular the ADBMS manifesto (Act-Net, 1996). For traditional ADBMS the active functionality is usually closely tied to the DBMS as a whole. This functionality was unbundled from ADBMS to be usable as an Activity Service in other contexts (Gatziu et al., 1998). This unbundled active functionality is a solid starting point (which requires extensions) for our current work. One step beyond this go approaches concerning ECA rule processing in distributed environments. In (Koschel et al., 1998) active functionality was extended into an ECA rule service for CORBA-based distributed, environments. Actually this approach is one initial step in our direction.

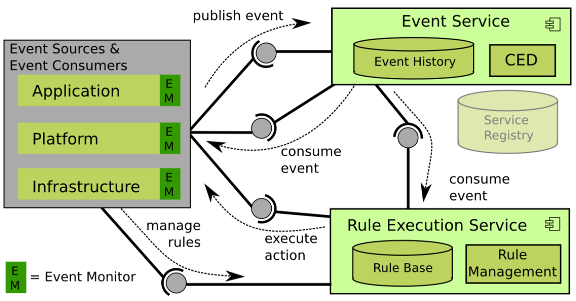

Figure 1: High-Level overview of the Activity Service architecture.

3 ARCHITECTURE

Following the unbundling approach from (Gatziu et al., 1998) we separate the Activity Service into different components (Figure 1). Each of the components provides one or more well defined interfaces with clear semantics. Thereby the concrete implementation of the different components is interchangeable. Furthermore the communication between the components is realized based on the concept of Service Oriented Architecture. The roles of the different components are outlined in the following paragraphs.

Event Service and Rule Execution Service

The event service provides all means necessary to be informed about the occurrence of events from the different event sources, to preprocess the events and to deliver them to the appropriate event consumers.

For each incoming event, it determines if there are event consumers that are interested in this particular event and delivers the event to them. A complex event detector (CED) evaluates the events and derives new complex events, which are fed back into the processing mechanism. To receive events from the event service an event consumer has to implement an appropriate event handler service, which needs to be published to the service registry. The event service discovers those services through the service registry.

The rule execution service (RES) receives events from the event service to evaluate them against sophisticated ECA rules. For the RES, the rules result in the execution of external action handlers that are provided by other third party components.

Event Monitors

The Activity Service suppports to obtain and process events from heterogeneous sources as typically found in heterogeneous information systems. We consider

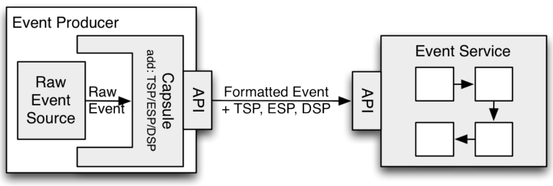

Figure 2: Capsule Concept.

on one hand 'active' sources like sources supporting triggers and callback mechanisms and sources with internal triggers. On the other hand we consider passive sources, which write all their actions into a log file, which needs to be actively monitored for changes. With the requirement to support several types of heterogeneous and highly distributed event sources we propose the capsule concept as a conceptual component. The capsule allows integrating event sources into our system and hides the raw event source that is producing the event information in a provider specific format. The capsule is responsible for (1) converting the event data format to the format used by the Activity Service and (2) to annotate the events with event, transport and domain semantic parameters as discussed in the next section. Figure 2 illustrates the capsule as part in the overall Activity Service architecture. The left hand part illustrates the capsule as part of the event sources residing as part of a cloud computing environment.

4 SEMANTIC PARAMETERS

Semantics describe the meaning of data and allow the right interpretation of this data. One design concept to introduce semantics to a (distributed) software system is the use of *semantic parameters*, which enable a system to deal with context-relevant information. The concept of semantic parameters has been extensively used in the definition of ADBMS. We utilize this well established definition as the foundation for our Activity Service which considers the following categories of parameters:

- **Transport Semantic Parameters (TSP)** describe how data is transferred and allow the usage of heterogeneous transport technologies as they appear in cloud environments. Simplified, TSPs allow in a technology independent way to specify all requirements that must be fulfilled by the underlying transport technology like for example the need for confidential event communication or the guarantee that events are delivered exactly once and in order of their occurrence.
- **Event Semantic Parameters (ESP)** describe the interpretation of events and their payload from a non-domain specific perspective. They describe general aspects of an event like for example the exact semantic of the given event timestamp or the lifetime of the event. These parameters are heavily influenced by the semantic parameters known from ADBMS.
- **Domain-Specific Parameters (DSP)** describe the meaning of events in the application domain and are not defined as part of the Activity Service but allow the delivery of domain-specific information within the event signaling.

In the remainder of the paper we will focus on the event semantic parameters and their link to the semantic definition of ADBMS.

4.1 Event Semantic Parameters

Event semantic parameters describe the non domain specific semantic context of an event. An event itself just contains some workload data like the temperature read from a sensor and some contextual information like a timestamp. However as the data needs some basic meta information to allow the processing without detailed knowledge about the event source itself as it was discussed during the introduction for the various meanings of a simple timestamp. For the Activity Service we aim to base the definition of this meta information on the semantic definition of ADBMS and the extensions towards distributed event based notification in (Koschel, 1999). In the following paragraphs we will discuss those parameters from the perspective of our Activity Service for cloud based environments.

The **signaling point** describes if an event was raised before the triggering state change happens (*pre*), after the state has already taken place (*post*) or the event replaces the actual state change by only giving this notification (*instead*). We consider all three parameters as a valid option with the change that for the signaling point *pre* we can not guarantee that given rules are actually triggered by such an event before the triggering activity is executed.

The **granularity** indicates the granularity of an event as it can represent a simple singular state change or an aggregation of multiple initial events or state changes, therefore having another granularity. From the perspective of the Activity Service we support the same granularities as they are defined for ADBMS: *instance oriented* where each single state change is considered as a single event and *set oriented* where multiple state changes are considered as one event.

The **net effect** indicates if the event triggering activity had any actual effect and is strongly motivated by transaction handling in database systems. As we are currently not supporting transaction handling, our model does not yet handle this parameter.

The **life span** defines how long an event is valid for processing. We consider the same two values (*implicit*, *explicit*) for the specification of the life span.

The **consumption policy** describes the order how events are processed. (Koschel, 1999) defined four policies. *Chronicle*, events should be processed by the order of the event creation. *Recent,* the last received event should be processed only. *Continuous*, the order of receiving is the order of processing (FIFO). And *cumulative*, events are processed as one whole group. We consider the same values but have a special focus on the details of ordered handling as it requires some effort in a distributed system where events are prone to arrive unordered. Therefore we expect some extension to the traditional ADBMS model.

The **coupling mode** is also one parameter for transactional behaviors in database systems. It indicates if the event happened within the transaction (*coupled*) or not (*decoupled*). It defines also if an event is thrown immediate or at the end of the transaction (*deferred*). In our current work we don't cover transaction handling explicitly and thus consider only the *deferred decoupled* value.

The **strategy** defines how the rule execution is triggered if multiple rules would be triggered by an event. The ADBMS semantic definition considers the following values: *parallel*, all matching rules are fired in an unpredictable order; *arbitrary*, one matching rule is picked randomly; *priority*, the rules have priorities and the rule with the highest priority is fired; *static*, a static order is given by an administrator; *dynamic*, the order is generated in runtime. In general we aim to support all of the available parameters, however with one important difference. As for a distributed event processing system it is usually the case that multiple rule execution components exist and a global ordering of the rule executions would be hard to achieve. Thus we consider the given attributes per processing component and not on a global scope. So on a global scope we only support the parallel strategy and allow a detailed specification per component.

4.2 Semantic Parameter Annotation

The Activity Service defines capsules as the glue between a raw event sender and the Activity Service components. Aside from their task to translate data formats, the capsule is responsible for the annotation of the generated events with the semantic parameters. As the capsule is specific to an event producer, it has the knowledge to assign the parameters. Thus the event source specific knowledge is encapsulated and the the Activity Service is based on the generalized semantic definitions.

5 CONCLUSIONS AND OUTLOOK

Current cloud services for event processing lack a well defined and vendor independent API and semantics definition, which results in a high risk of a vendor lock-in. With our Activity Service concept we address these challenges by providing a generalized event communication and processing service that can bridge the gaps between multiple provider specific proprietary services. It introduces a solid semantic definition based on three categories: *transport*, *event* and *domain* semantic parameters. The discussions showed that the semantic of ADBMS can be used as foundation for the event semantic parameters but also showed the importance of the transport semantic parameters due to the distribution in a cloud environment. The next step will thus be the definition of these parameters. Our current implementation efforts are for the moment mostly focused on overcoming technological hurdles but based on our prototype we will realize the aforementioned semantic parameters to provide working system that can be used to evaluate the concepts for real world scenarios.

REFERENCES

Gatziu, S. and Koschel, A .et al.,1998 , *Unbundling active functionality*. In ACM SIGMOD Rec., vol. 27, no. 1, pp. 35–40.

Goyal, P. and Mikkilineni, R., 2009, *Policy-based Event-driven Services-oriented Architecture for Cloud Services Operation and Management*. In Proc. 2009 IEEE Intl. Conf. on Cloud Computing.

Koschel, A. and Lockemann, P. C.,1998, *Distributed events in active database systems: Letting the genie out of the bottle*. In Data Knowl. Eng., vol. 25, no. 1-2.

Koschel, A., 1999, *Ereignisgetriebene CORBA-Dienste für heterogene, verteilte Informationssysteme*. Dissertation, Universität Karlsruhe, Germany.

Paton, N. W., editor, 1999, *Active Rules for Databases*. Springer, New York.

Schaaf, M., Koschel, A., Gatziu, S., Astrova, I., 2010, *An Active DBMS Style Activity Service for Cloud Environments*. In Proc. of the 1st Intl. Conf. on Cloud Computing, GRIDs, and Virtualization. Lisbon, Portugal.

Schaaf M., Koschel, A., and Gatziu, S., 2011, *Event processing in the cloud environment with well defined semantics*. In 1st Intl. Conf. on Cloud Computing and

Services Science, SciTePress, p. 176-179, Noordwijkerhout, Niederlande.

The ACT-NET Consortium, 1996, *The Active Database Management System Manifesto: A Rulebase of ADBMS Features*. In ACM SIGMOD Record, 25(3):414–471.

Widom, J. and Ceri, S., editors, 1996, *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann Publishers, San Francisco, USA.

Wishnie, G. and Saiedian, 2009, H., *A Complex Event Routing Infrastructure for Distributed Systems*. In Proc. 33rd Int. Conf. IEEE COMPSAC.