

Architectural Issues for Cloud Computing Applications

Marc Schaaf, Stella Gatzju Grivas

University of Applied Sciences NW Switzerland

marc.schaaf@fhnw.ch, stella.gatziugrivas@fhnw.ch

ABSTARCT

In this chapter we focus on cloud application architectures, as the availability of suitable architectural concepts is imperative for the creation of cloud applications. For this purpose we will discuss several application areas of cloud computing to outline the requirements that a cloud application has to fulfill. Based on those requirements we will then discuss an architectural approach for an event driven architecture that allows the implementing application to benefit from the cloud environment. To demonstrate some of those concepts in a more practical manner, we shortly discuss the realization of an Activity Service as a specialized middleware that realizes our architectural concept to ease the development of event driven applications in the cloud. Based on the description of the event driven cloud architecture and its realization we will give an insight into the special conditions of the cloud environment an architecture has to deal with.

MOTIVATION FOR CLOUD COMPUTING ARCHITECTURES

Current cloud architectures from both, the commercial and the academic side, follow a broad variety of approaches. On one side, there are the well-established market players such as Amazon, offering single component IaaS services and Microsoft or Google with their PaaS services. On the other side, there are the theoretical and yet to be fully implemented basic architectures covering a single layer (e.g. Open Cirrus [1] and Reservoir [2]) or multiple layers (e.g. CloudIA [3] or CCOA [4]) of the cloud. However concrete guidelines for the realization of an application architectures for cloud environments are currently missing.

Most researchers and professionals agree on the typical cloud stack defined by [5] which can be seen as a basic architecture for cloud computing (Figure 1). Following the paradigm “everything-as-a-service” (XaaS) the stack comprises the different services types. Each service type (IaaS, PaaS, SaaS and HaaS) represents a layer in the stack. The layer can be further divided into different sub-layers. Every layer can profit from services provided by any of the other layers below. However it is important to note that the stack is currently to be seen and used as building blocks for cloud services. It cannot be considered as an architectural model that can be used to design cloud applications. The different layers have different requirements and serve different purposes especially with regard to the requirements that have to be handled by a suitable architecture. Even though Cloud Computing promises huge benefits regarding the agility and with this, the usage of everything as a service, there is a great complexity in realizing and integrating cloud applications. Depending on the layer on which the obtained cloud services reside, the aspects, a consumer has to worry about, change. For example the usage of a SaaS offering from the cloud implies that the service provider has to handle most of the classical problems of scalable distributed applications. On the other hand, the usage of an IaaS offering implies that the consumer (the cloud user) has to handle the distribution and scalability aspects of his application. For both layers PaaS and IaaS, the cloud application needs to be based on a suitable architecture to allow the consumer to benefit from the cloud environment. Thus the suitable architectural concepts, frameworks and guidelines are required to allow the consumer to build cloud applications. In this chapter we focus on architectural aspects for the IaaS layer. However the most of the descriptions are also valid for the PaaS layer with the exemption that the provided platform already dictates or realizes several architectural aspects. We have analyzed application domains like opinion mining, disaster management, smart grids and online business platforms which all require scalable architectures. For example in opinion mining, millions of web sites have to be crawled in (near) real time to detect all the potential opinions expressed online; the software architecture has thus to be carefully designed to enable rapid text analysis, opinion extraction and data visualization [6]. In disaster management systems an applica-

tion architecture has to provide an adaptable application that can rapidly scale up in disaster situations and handle partial failures within its usual communication and processing systems. For the management of smart grids, multiple largely distributed and heterogeneous information sources need to be monitored to allow an efficient management of a nationwide energy production, distribution and with smart grids also the consumption. To allow those management tasks in an efficient manner, real-time statistics and forecasts of the energy production and consumption is needed which requires huge amounts of calculation resources and a tight integration with the smart grid monitoring and regulation information systems.

To fill the aforementioned gap in the availability of suitable application architectures, we work on a novel approach of an architectural concept. The Basis build the mentioned application domains from which we extract the requirements for building applications that allow the particular domain to benefit from the cloud environment. Our approach follows the architectural concept of Event Driven Architectures (EDA) so that components communicate with each other via events which are sent via an appropriate communication middleware to all subscribers who need to act on them. The other components decide themselves for which type of events they will subscribe and on which they will respond. As one of the aspects of Cloud Computing is the heterogeneity of the provided services between the different providers, we furthermore provide an abstraction from the heterogeneity of the underlying communication technologies and the provided infrastructure, by using the principles of Service Oriented Architectures (SOA) [7]. Furthermore we support Complex Event Processing (CEP) for analyzing low-level events to discover event patterns (e.g., by aggregating them or combining them in a specific context: time, space, semantics) and to act on them in an efficient and timely manner [8].

In this chapter we will first introduce the mentioned application domains for cloud computing then derive some general requirements and based on them discuss our architectural approach as an example for coping with the additional requirements. To round up this description we will then discuss parts of implementation of the architectural concept in the OM4SPACE project [10] where an activity service with a precisely defined Active DBMS style ECA rule and execution model for the cloud (as reported in [11]) is developed. The activity service will bridge the gaps between several cloud specific messaging or notification mechanisms by providing an abstraction across the vendor specific API's.

DESCRIPTION OF APPLICATION DOMAINS FOR CLOUD COMPUTING

The following descriptions roughly outline four application domains for cloud computing that we encountered during our research work. Each of those domains requires a particular set of requirements. However it will become evident that certain requirements can be found in all domains. From those domain requirements we derive the general requirements for cloud computing architectures in the next section.

Opinion Mining

Opinion mining deals with the increasing need of decision makers to capture the public opinion about their products or actions. Several systems have been developed to automatically extract and interpret Web content related to a specific concept or topic. However, today opinion extraction software is mostly relying on a small list of potential data sources to support the analysis of a restricted set of topics. The success of these small/moderate-size systems shows the potential feasibility of more global large scale opinion mining systems on heterogeneous data sources that would be able to analyze the opinion of the population on some general questions, such as counting the people having money issues, knowing what the population thinks about the latest news events, etc. Some recent studies demonstrated that such systems could potentially replace, or at least complement, traditional polls.

The architecture of an opinion mining system has to be scalable: millions of web sites have to be crawled in real time to detect all the potential opinions expressed online; the software architecture has to be carefully designed to enable a real time text analysis, opinion extractions and data visualization. One of the key challenges is that the natural language processing workflow is highly interconnected due to implicit or explicit user feedback. This means that every time a new entry is added in the source database, a large amount of text understanding procedures (and in particular cross-document co-references) has to be launched again. A naive brute-force reset of the databases is naturally infeasible and smarter ways of updating the extracted content have to be found.

Such systems do not exist today but due to the dynamic availability of huge resource amounts as they are provided by cloud computing providers, they can now be realized. However they need to be based on a suitable infrastructure to handle the huge amounts of information sources and the need for rapid analysis as well as the capability to scale based on the current user feedback that needs to be gathered and processed. The architec-

ture of an opinion mining system has to realize the following requirements to allow the opinion mining process to benefit from the cloud environment:

Interoperability

The whole opinion mining process has to act on largely distributed and heterogeneous information sources. Furthermore such an application is likely to span across several cloud providers which add further to the distribution and heterogeneity. Despite those facts all application parts need to be able to communicate with each other and need access to centrally stored pieces of information. For this an efficient communication mechanisms that bridges across the gaps created by the provider and the information source heterogeneity but still remains easy to realize and to use is required.

Adaptability and Flexibility

Opinion mining applications need to detect changing opinions quickly. For this it is required, that enough processing resources are available to process all the gathered information pieces in a timely manner. However for example during an election period or after some public discussions, much more data has to be processed than usually. In these scenarios the cloud environment can provide the additional resources on an as needed basis. However this requires that the application architecture is capable of integrating and removing the additional resources dynamically to take advantage of the environment. Another important aspect is the requirement for dynamic adaptation of the processing flow and the information that needs to be gathered. This adaptation needs to be supported by the architecture for a largely distributed system running on various heterogeneous platforms of different providers.

Security and Integration

Even though opinion mining applications derive their knowledge from publicly available sources, their processing results can be seen as critical and need to be secured in a special way. Thus the usage of a combination of a public and a private cloud is a likely scenario alongside with the possibility that only the first processing steps take place in a cloud environment and the later steps, including the simulations, are realized by foreign applications. Thus the application architecture needs to support integration of application parts in private clouds and “normal” datacenters.

Disaster Management

Disaster management is a complex multi-dimensional process involving a large number of interoperating entities (teams of humans and systems) and is affected by various social, medical, geographical, psychological, political, and technological factors. From the information technology viewpoint the disaster management processes can be hindered by lack of adequate, comprehensive and timely information; the presence of conflicting goals, policies, and priorities; lack of effective coordination between different rescue operations augmented with inability of many units to act autonomously [28] An Integrated Disaster Management System should be able to create the necessary interactions between the involved systems which are dispersed and create the need functionality. We have identified the following requirements must be fulfilled by an integrated disaster management system.

Interoperability

The disaster management system uses data origin in several heterogeneous distributed systems. There is data about person both about victims and about relief personnel; data about damages to buildings, infrastructure and belongings; weather data; geographical data about roads and other landmarks as well as satellite imagery that can be used for damage assessment; logistics data about vehicles, delivery times, etc.; communication and message data; financial data needed to manage the collection and distribution of donations; data in blogs; news reports; etc. All this data must be collected and automated pre-processed (the variety of semantic and syntactic heterogeneity must be captured) to be used for decision making during the disaster management An integrated disaster management system presumes that information gathered in one phase is available to all next phases. For example information gathered in preparation phase is critical for responding a disastrous event. However it is valuable for recovery phase, as data collected in preparation phase can be used by the decision makers in recovery action and risk reduction for further events.

Accessibility and Security

All participated entities (systems, organization or the public user application) should be able to have access to the integrated disaster system. This also raises the question of information security. Access policy should also be established based on the pre-defined policy for information sharing.

Adaptability and Flexibility:

Incident command system requires fast flexibility and adaptability, on the fly, and effective data integration from multiple subsystems. The prerequisite is the ability to change something in accordance to the situation. When unpredicted scenario happens, for instance a terrorist incident, the system may need to include information from National Guard or Governmental agents.

Geographic information capability

All the information captured stored and analyzed should be able to be presented with a reference to a geographical location data. Integration of GPS and Satellite images with situation report will give a border view of the disaster at hand. Resource can easily be located and dispersed. Victims can receive faster response. Such capability will also improve the preparedness and recovery phases.

Reliability

“Responding to disaster situations, where every second counts, requires reliable, dedicated equipment. Public safety officials cannot depend on commercial systems that can be overloaded and unavailable; experience has shown that these systems are often the most unreliable during critical incidents when public demand overwhelms the systems”.

Smart Grid

Smart grid can be also described as information-enabled grid. Information is collected through existing, new and emerging sensing and measurement systems. The information may be used to improve efficiency, reliability, and cost effectiveness of power system operations, planning, and maintenance. From the ICT perspective, this increased information flow is a central aspect for the realization of smart grids. To allow grid management systems to optimize the grids efficiently and to overcome power outages rapidly, they need just in time access to the information that is produced by the different grid participants. However, providing access to the huge amount of information from a country wide power grid, can be considered as a difficult task at best. Thus it is imperative to base smart grids on an appropriate information processing architecture.

The complexity of the required data and its processing becomes more evident when the huge amounts of sensors and actors within a country wide energy distribution system are considered. For example a smart grid, could be separated into different zones where each zone would cover a state. Each of those first level zones would be subdivided into several second level zones which could represent cities and their suburban areas or rural districts. These zones can once again be broken down until they are at the level of streets and households. These different zones form a hierarchy which can be used by the grid monitoring and management.



Within each of the low level zones, several sensors and actors have to be monitored and controlled. For example x each household within a district could be informed about its current energy consumption. Furthermore, each household would be interested in the information when it would be the cheapest to obtain energy to for example turn on a dish washing machine or a washing machine. A local grid management component could combine the information from the different households to detect that there currently is an overproduction of energy. To avoid wasting this energy, it could notify some of the households that now would be a good time to

consume additional energy. In the opposite case, where more energy is consumed than available, a controller could activate local in hold power generators. Furthermore the local need for additional energy should be populated to the next higher management zone. Once a management system in this zone detects the additional energy requirements it could activate a storage power plant and notify the lower zones to shut down the in house generators. This example outlines the huge amounts of sensors and actors as well as the information flows that need are required to allow the intended amount of agility in smart grids.

In addition to the local reactive management of the energy consuming and producing devices, a central management is required which is capable of enforcing country wide policies and decisions. Furthermore simulation mechanisms are required to provide reliable forecasts for the energy consumption as well as for the production (e.g. Solar power plants). Those central management mechanisms are perfectly suited for cloud computing based infrastructure where they can be rapidly deployed and scale as needed. Furthermore they have extended requirements regarding the reliability and security of the underlying cloud infrastructure. Another very important issue is the integration with the information sources in the distributed sensors from all over the power grid as well as legacy systems. The important requirements for a cloud-based smart grid information management architecture can be summarized as follows:

Integration and Interoperability

The central aspect of a smart grid information system is the integration of various largely distributed and heterogeneous information sources and information consumers in an efficient and scalable fashion. In particular this requires that the corresponding cloud application architecture supports the integration of the information sources in in a dynamic and flexible fashion. Furthermore legacy systems like power distribution controlling or power production and consumption forecast systems need to be usable within the cloud based system while avoiding a tight coupling between the different communication protocols and services. It is also quite likely that the smart grid information system spans across several cloud providers and cloud types like private and public clouds. Thus once again the efficient and interoperable integration with cloud provider/platform specific services is an important requirement.

Security and Confidentiality

As already mentioned it is likely that a smart grid management system spans across several cloud types. One reason for this heterogeneity is the extensive requirements regarding security and confidentiality. The processing system and the used environment must assure that no unauthorized access to the power grid management is possible. Furthermore the handled data needs to be protected as it allows to gain access to private customer data related to their power consumption patterns and thus their consume behavior.

Reliability, Adaptability and Flexibility:

Alongside with the protection against unauthorized access, the protection against failures and system downtimes is an important requirement for a possibly country wide power grid management system. For this purpose the grid information system architecture needs to cope with partial failures in a graceful way by for example implementing fallback strategies like migrating parts of the system to a different cloud provider or re-routing information flows through other communication channels and services.

In addition to the possibilities of partial failures or even denial of service attacks, the system architecture need to be able to handle huge quantity of (sensor) data that occur during normal system operation. A cloud computing based architecture should be capable of adapting the actual usage of resources dynamically to take advantage of the cloud environments agility.

Click Behavior Analysis of Online Shop Users

Modern online shops need to scale rapidly in certain peak load situations like after a successful advertisement campaign or before Christmas. Cloud Computing services can be used to provide the required infrastructure based on IaaS dynamically. However this of course requires that the used web shop software and all linked software systems can handle the adding of additional resources to the processing system. For example in the case of PHP based web shop that uses a MySQL database server as its storage backend, adding additional web servers that execute the web shop software can be added. However in this case it is also required to introduce a load balancing component in front of the web servers to distribute the user requests also across the new servers. Thus such a load balancing and sharing of the backend storage system needs to be supported by the web shop and its architecture.

Nowadays concepts have been developed to enrich web shops or in general web sites with dynamic recommendation and adaption systems that support the visitor in finding what he is looking for or what the shop owner wants him to see. One approach to realize such systems is the use of click stream analysis. In this case

the users requests (each of his clicks on links in the website) is analyzed to build up a user profile that represents his interests and his behavior. If such an analysis is aimed to change the website content dynamically while the user is still browsing through the website, the required calculations need to take place in near real time. Thus it is required to provide massive amounts of resources to such a processing system in peak load situations to guarantee for the aimed user experience. Of course those resources can be provided by cloud computing services if the architecture of the web shop as well as the processing system can take benefit from the dynamic environment. In summary, the requirements in the following areas are particularly relevant for a cloud based application architecture:

Integration

The recommendation system needs to be integrated with the CMS that provides the usage details from the website visitors and requires the timely feedback of suitable recommendations. For this the application architecture needs to support suitable mechanisms that can be adapted to utilize communication mechanisms available in the currently used cloud environment. Furthermore the different parts of the recommendation system (e.g. interest detection, profile management, recommendation rule execution) need to communicate with each other in an efficient and suitable way. Thus the architecture should allow not only the decoupling from the communication mechanisms used for external communication but also for the communication between the application components. This will allow the recommendation system to benefit from highly optimized cloud provider specific communication services while staying independent of the concrete provider to avoid a vendor lock-in.

Adaptability and Flexibility

As the load of the monitored online shops vary the load of the recommendation system also changes based. In a cloud environment the resources are available dynamically, thus the application architecture needs as in all the other scenarios be able to handle the dynamic of the available resources. Furthermore a recommendation system has to deal with changing rule sets as the owner of the system will adapt the recommendation rules over time to suite more precisely or to adapt to changing customer needs and interests. Every time the rule set is changed, a reevaluation of the gathered data needs to be done to allow the system to generate the recommendations based on the new rules. Their reevaluation needs huge amounts of calculation resources and should be achieved in a reasonable time frame to allow the system to be as agile as required by the owner. For those cases the application architecture needs to be able to utilize huge amounts of additional resources in an efficient way for a short period of time.

Security and Data Confidentiality

A recommendation system gathers information on a given website visitor to derive his interests and behavioral patterns to generate suitable recommendations. Even though this data alone is stored in an anonymized form and can only be linked to a concrete user by the CMS, the data should still be considered as confidential and needs to be protected. Furthermore the system should be protected against unauthorized manipulations of the user profiles to for example prevent an attacker to trigger the generation of an additional discount that should only be available to some special customers. As a result, the application architecture is required to provide a suitable amount of security for 1) the communication between the application components and the CMS and 2) the stored data in the cloud providers data storage services.

SUMMARY OF THE REQUIREMENTS FOR CLOUD COMPUTING APPLICATIONS

In the outlined scenarios of the several different application domains, various requirements were stated. This section now sums up those requirements in an abstracted form to provide the foundation for the following architecture discussions.

Agility and Scalability

All outlined scenarios have in common that they expect increased agility from the usage of cloud based services. Commercial cloud solutions usually advertise with exactly this increased agility and flexibility which they promise to be available to the customer with just a few clicks. However such advertisements often neglect the need for additional tasks alongside the usage of cloud services like the integration into the existing process and system landscapes as well as the data migration. Thus solutions and guidelines are required for use cases of complete implementation scenarios, starting with the selection and ordering of cloud services and continuing to an up and running cloud system and its maintenance. Furthermore the scenarios assumed the possibility to scale their application dynamically by using cloud services. However as for agility in general, an application architecture needs to handle the horizontal scaling that is usually employed in cloud environments.

Thus cloud architectures should build upon the concepts and patterns that have already been developed for scalable distributed applications and adapt them to the increased agility of the cloud environments.

Interoperability and Integration and the Risk of a Vendor Lock-in

Interoperability has a different meaning on each layer of the cloud stack. From an architectural point of view the interoperability of single services and the seamless interoperability of computational and storage resources and the communication services that are used for the communication with other application components inside and outside of the cloud can be distinguished. The scenarios where often based on the usage of several different cloud environments and types from different providers. Furthermore the integration with other, non- cloud, application like legacy or in-house applications is often a requirement. Thus standards within each layer as well as for cross-layer interoperability have to be defined to allow interoperability across the borders of a single cloud environment. In addition approaches for the integration with existing enterprise architectures are required. The resolution of the mentioned interoperability problems are also one of the major risk that comes with cloud computing: The risk of a vendor lock-in due to the usage of proprietary services or service combinations. Furthermore the cloud management services are often very proprietary and limited to a vendor or a specific cloud service. Managing and monitoring of multiple inter-operating cloud services is thus an open issue for cross-cloud management and monitoring solutions which are required for large scale applications.

Security and Reliability

All of the outlined scenarios process sensitive information at some point that needs to be protected against unauthorized access. Furthermore most applications have the requirement that the processed data is not manipulated and that the integrity is ensured even if the application is distributed across several cloud providers and possibly even across several countries. As a result security aspects must be handled not only on the cloud provider side but also on the cloud application side. Thus security aspects need to be covered by cloud architectures on both the provider and consumer side. Alongside with the confidentiality and integrity of the processed data, the availability and reliability of the cloud application is an important point in all of the scenarios. However the reliability of commercial cloud services is usually high and often is granted through SLAs. The aspired federation of clouds and combinations of different cloud services raises new questions regarding SLA management and reliability assurance. Furthermore aspects like network access have to be taken into account. Thus availability and reliability aspects also need to be reflected in the cloud application architecture especially when the application has to act across the borders of a single cloud where it should for example be able to handle partial failures of the communication of processing systems.

USING EVENT PROCESSING FOR BUILDING ARCHITECTURES OF CLOUD COMPUTING APPLICATIONS

One major concept behind Cloud Computing is its service orientation. In a computing cloud, everything is provided as a service, starting with the infrastructure, continuing with the platforms and finally the complete application software. From the architectural level of view the building of real domain applications which use cloud services usually follows the concepts of service oriented architectures (SOA). A cloud computing application must be build based on services with clearly defined interfaces and semantics to allow a strong encapsulation and decoupling of the components. These services can be on-premise or cloud services.

SOA based applications feature a relatively easy possibility for the recombination of the application services to reflect new processes. But the limitation to sequential and linear processes remains, due to the static behavior of the application which is defined at design time.

The architectural model of Event Driven Architectures (EDA) [12] conquers this limitation. It allows building applications to handle unpredictable processes because the process flow of an EDA application is determined at run-time, triggered by events. EDA applications are able to respond on casual events occurring in random order. While SOA is related to linear work flows, EDA fits for asynchronous long-running processes and to handle deterministic unpredictable workflows.

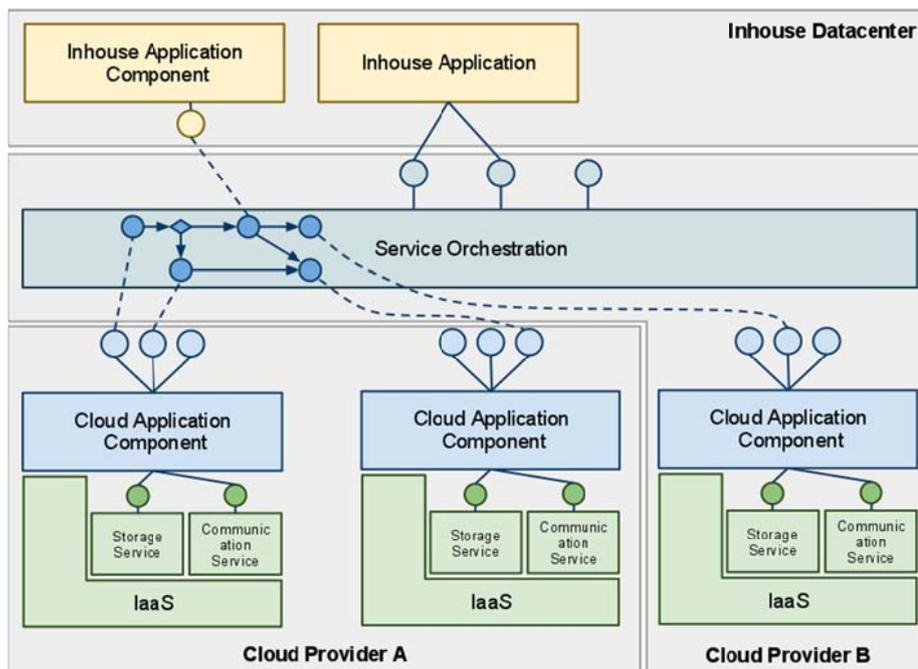


Figure 3: A possible constellation for a Cloud based Service Oriented Architecture

Both architectural concepts, SOA and EDA, reduce the complexity of applications dividing them into modular, reusable, encapsulated and loose coupled components. They are service-oriented and message-based sharing a common semantic. SOA provides a reliable and flexible integration concept based on mature standards while EDA adds the capabilities to handle unpredictable workflows, responding in real-time on random events or event-streams. In combination they match perfect to build cloud applications since they are well suited for implementing distributed applications across multiple platforms and enterprise boundaries. The new paradigm, also called Event-Driven-Service-Oriented-Architecture (EDSOA) [21], has been already investigated in the literature [referenz]. It is a modular architecture using asynchronous messaging to interconnect its components and its components are capable to respond autonomous on events, detect event-patterns or process event-streams. Going more in depth in the area of Event Driven Architectures we realize that event processing is the underlying mechanism of EDA. Event processing has been extensively investigated in the 1990s starting with the approach of active databases supporting ECA-rules [11]. Relevant single primitive or complex events (mostly database operations) have been detected and signaled by the active database and as a sequence actions have been executed. The next generation are the event notification systems monitor relevant systems, which produce an event sequence (event stream) and notify subscribed clients upon the event occurrence (push/pull mechanism. In EDA, an *event* is the center of the communication and is used to control the components which act autonomous, and are fully decoupled. Only the semantic of the events must be understood by all components. The core principle of event processing is that events are sent via a communication middleware to all subscribers who act on it. The other components decide themselves which types of events they will subscribe for or on which they will respond. For example the components could select which type of events they subscribe to or on which event types they respond based on a business rules framework. Event Driven Architectures are tightly linked to Complex Event Processing (CEP). CEP is an emerging enabling technology to apply context-aware knowledge from and against large amounts of event data in near real-time. CEP engines can process multiple streams of event-oriented data to identify meaningful events in real time, based on the business rules framework. They analyze low-level events to discover event patterns (e.g., by aggregating them or combining them in a specific context: time, space, semantics) and to act on them [8, 13, 14]. The huge amount of events to be processed is thus reduced within several processing stages. Building on semantic technologies (e.g. on Ontologies like OWL-S) there is the prospect of combining event processing and semantic technologies into the area of Semantic Complex Event Processing (SCEP). The event processing engines can understand what is happening during the processing of the event streams which is relevant to the events and (process) states. Semantic event processing uses for the signaling of an event the knowledge base (e.g., defined as ontology modeling of the relationships between the different events) and can also derive knowledge which is not explicitly described in the event definition. SCEP-enabled event processing engines will have (1) a description of what is happening in terms of events and event patterns based on

situations and process states (like in CEP), as well as (2) a higher-level plan for the (re)actions and activities they can invoke, which can lead to (monitored) follow-up events.

Current research [15] includes rule-based Event Processing Languages (EPLs) for the Web, such as Reaction RuleML, which employs reaction rules that have evolved from existing rule-based technologies such as Production Rules and Event-Condition-Action (ECA) rules. They use SCEP-generated facts and knowledge to derive further decisions and trigger automated reactions. Moreover, such EPLs can exploit the declarative expressive power of semantic rules as a means to specify knowledge in a way that is understood by 'the business' and is executable by CEP rule engines.

First ideas for semantic event processing [16, 17] focus on one common super event ontology which generalizes and ignores the differences of the various application domains. First approaches for a specific domain, application and task ontologies is the work of [18] which propose modular sub ontologies for stock applications and the work of [19].

In this chapter we present an architectural concept for event driven applications as we consider event processing as a particularly suited approach for cloud applications. Compared with the derived requirements from the previous sections, an event processing architecture features the following attributes:

Agility and Scalability

Event processing application components work on the processing of mainly self-contained messages (events) which results in the creation of new self-contained messages. In general terms this allows the easy distribution of the processing components across several virtual machines in possibly even several clouds. In practice each of those processing components will not be able to act completely independent of the other and without interaction with some centralized data repositories. However the general concepts of the architecture already support the distribution which makes it easier to find suitable solutions for the remaining challenges. With the integration with semantic technologies the agility of the application can be further extended as automatic management components can adapt the processing system easier due to their gained knowledge of the managed application.

Interoperability and Integration and the Risk of a Vendor Lock-in

Most parts of an event processing systems rely on a simple communication principle: receiving and sending messages/events. This communication mechanism can be ported to a lot of communication technologies while preserving the semantic and hiding the technical facts from the components to allow them to be provider independent. This of course doesn't cover access other cloud services like for example to storage services. However as the following description of an event based cloud architecture shows, solution can be found fairly easy.

Security and Reliability

The proposed event processing based cloud architecture features now extensive concept to ensure security in terms of data integrity and confidentiality. However as the whole system builds upon the notion of event based communication, the communication system can be used to ensure required security level. With regard to reliability of the application, the usage of a central monitoring and management system together with an ontology based description of the applications components and communication links, (automatic) measures can be implemented to assure component redundancies and automatic failover if components fail or overload.

DEFINING THE ARCHITECTURAL OF AN OPINION MINING APPLICATION AS A CLOUD COMPUTING APPLICATION ARCHITECTURE

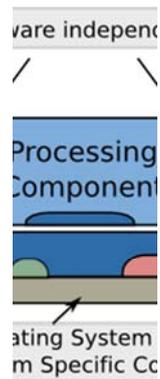
In this part we discuss how our architectural approach towards the usage of event processing in the cloud applications can be used in the domain of opinion mining. For an opinion mining system massive amounts of information need to be processed from various information source scattered all over the internet. Furthermore the information amount that needs to be processed is likely to vary over time. For example during an election period, the single statements of the politicians can sometime trigger massive responses cause a burst in the amount of information that needs to be processed in a given time window to be useful. As a result an opinion mining application can benefit from cloud computing resources to allow the dynamic scaling of the information gathering systems during such bursts. Furthermore they benefit from the dynamic event driven processing as outlined in the previous sections.



The actual processing of the information needs to be done by several specialized components (e.g. Natural language processing solutions). A possible organization of the components for the opinion mining on a large scale is given in Figure 4. The first processing stage involves gathering and pre-screening raw content and is located in close proximity to the actual content sources and possibly distributed across geographic regions (e.g. in different data centers from different cloud providers). The second processing stage is located at a “nearby” cloud provider and works with aggregated and filtered content that is processed for opinion mining and sentiment analysis. The third stage of the system can be located in a private (secure) cloud and does the final processing and thus produces critical results that have to be secured against unauthorized access. The results from this stage can e.g. be feed into simulation or visualization engines.

The processing of the information pieces is done mostly independent of the other processing flows. Only in the final stage of the processing system the data is aggregated into the required results. However the processing components need access to a central information repository which contains details on the information that is relevant for the monitored opinions. As this data needs to be changeable rather quickly to adapt the system to new situations (e.g. new aspects for the monitored topics) the data can't just be distributed together with the component deployment. To allow easy access of the processing components to the central repository without the need to cope with details of the underlying heterogeneous communication technologies, a run-time container is defined which provides access to the data repository (Figure 5).

In addition to the abstraction from the data repository access, the container provides access to the event based communication middleware in similar means. It thus hides the middleware specific details from the processing component and in addition takes care of the information routing between the different components (in cooperation with the middleware and a management component). The encapsulation of the communication details is even more important in the environment of cloud computing as it already is for “normal” distributed application. This increased relevance is caused by the various service offerings of the different providers. For example Amazon offers with their Simple Queue Service (SQS) a special message based communication service that is optimized for their environment seems thus more suitable than using a normal enterprise messaging system that would also be managed and scaled in addition to the application components. However SQS is only available in the cloud environment of Amazon. For a largely distributed application it is however likely that some application parts are deployed with other cloud providers or in a private data center where another communication mechanism is available. Furthermore the communication between different clouds or data centers needs additional features of the communication middleware like data encryption or extended reliability mechanisms that would slow down the communication within the borders of a single data center. Thus an important aspect for cloud computing applications is the abstraction from cloud provider specific services.



For the specific case of event based communication and processing, the p OM4SPACE project aims to provide a general communication service w specific services while providing an ADBMS based processing semantic One central aspect for of cloud computing is its extended agility which al as needed and release them as soon as they are unnecessary. This allows a ically at varying loads but also to reduce the fees that need to be paid. Wt service as it is used in this application scenario, the usage of computing r ly. Instead the application that is deployed on the cloud has to take care o quiring additional resources and integration them with the application.

The automatic scaling of the here presented event driven application is he component that monitors the different event processing components with The management component holds a model of the processing application the requirements of the different components as well as dynamic informa ponents and used communication systems. It uses this model together wit cide if new resources (in the form of additional virtual machines) need to cute those scaling actions, the management system has to feature provide ing cloud service APIs. Via such an adapter the management component c the corresponding cloud.

In addition to the acquiring and initialization of the new resources, the m event routing between the components to integrate the new or exclude the ccessing system. The dynamic adaption of component routing is once agai time container which allows the transparent rerouting of the communicati Such a dynamic staged infrastructure reflects the principles of cloud com| ccessing system is located in public clouds where a rapid scaling based on stages are located in private clouds where privacy is ensured and access n sharing of resources to adapt to changing loads.

CONCLUSION

Data and processing intensive applications like smart grid, disaster manag in heterogeneous distributed environments and are very good candidates f ever getting real advantages from cloud opportunities like scalability and presumes that the applications follow suitable architectural models and pr requirements that arise from the usage of a Cloud Computing environmer digm of Event Driven Service Oriented Architectures (EDSOA) we gave how data processing in opinion mining takes place and how it can benefit Service Offerings. The discussions of the application architecture require out the major open issues and the challenges. Our discussions focused on between the application components and foreign systems. Other open asp issues and aspects of data data storage. A solution could be a layered syst usage model.

REFERENCES

>> Literature list is currently outdated! Will be updated for the ne

- [1] Avetisyan, A.I, et al. (2010). Open Cirrus: A global cloud computing test bed. Cor
- [2] Rochwerger, B. et al., 2008. The RESERVOIR model and architecture for ope Research and Development 53(4).