HORIZON2020 FRAMEWORK PROGRAMME

TOPIC EUK-03-2016

"Federated Cloud Resource Brokerage for Mobile Cloud Services"

D2.1

State of the art and Requirements Analysis

Project acronym: BASMATI

Project full title: Cloud Brokerage Across Borders for Mobile Users and Applications

Contract no.: 723131

Workpackage:	2	Complete Analysis, Architecture, Integration
Editor:	Netsanet Haile	SNU
Author(s):	Jörn Altmann	SNU
	Ana Juan Ferrer	ATOS
	Antonia Schwichtenberg	CAS
	Baseem Alathwari	SNU
	Azamat Uzbekov	SNU
	Emanuele Carlini	CNR
	Jamie Marshall	AMENSIK
	John Violos	ICCS
	Young-Woo Jung	ETRI
	Lara López	ATOS
	Patrizio Dazzi	CNR
	Richard Wacker	CAS

	Enric Pages	ATOS
Authorized by	Konstantinos Tserpes	ICCS
Doc Ref:	D2.1	
Reviewer	Konstantinos Tserpes Patrizio Dazzi Emanuele Carlini	ICCS CNR CNR
Dissemination Level	Public	

Document History

Version	Date	Changes	Author / Affiliation
v.0.1	03-08-2016	Created ToC	Netsanet Haile / SNU
v.0.2	05-10-2016	Revised ToC	Netsanet Haile / SNU
v.0.3	17-10-2016	Topic assignment	Netsanet Haile / SNU
v.0.4	10-11-2016	First Draft	All partners
v.0.5	20-11-2016	First version	All partners
v.0.6	23-05-2017	Review version	All partners
v.0.7	08-06-2017	Final version	Netsanet Haile / SNU

Executive Summary

This deliverable examines the state-of-the-art solutions, both focusing on the scientific literature and on the existing, cutting-edge, technologies that relate to the activities that will be conducted in BASMATI project. Along with the presentation and discussion of the existing approaches and solutions, this report describes possible future trends and presents the project requirement analysis, both the one considering the use cases technical requirements as well as taking into account the technology insight coming both from linked research initiatives and interactions with innovation experts.

This document is essentially aimed at a first snapshot resulting from the activity aimed at the identification of research paths and potentially useful approaches. The idea for this report was to comprise a “live document”, continuously updated by the consortium members. Changes were incorporated during the first year with the purpose to include new interesting technologies and solutions that were proposed and released. In fact, during the whole course of the project, the technologies and requirements related to BASMATI will continue being investigated in order to ensure that the objectives and innovations of the project are valid, work is performed taking into account the latest state-of-the-art and developments fulfill the identified goals and requirements. However, this stable version of the document will serve as a guide for the baseline technologies and initial requirements upon which BASMATI builds. As such the report comprises a central document that drives the project processes and it is directly linked to the project vision implementation.

Table of Contents

1 Introduction

Cloud Computing has transformed the IT by providing computing and storage as on-demand utility services according to the pay per use model. Large IT behemoths provided their spare resources for renting to other private enterprises and individuals. With the evolution of services and applications that can be brought on the cloud, a single cloud solution cannot provide the heterogeneity and functionality required for many business solutions. Therefore, the initial concept of cloud computing has rapidly evolved into multi-cloud environment, which gathers together multiple and heterogeneous cloud datacenters and service providers.

The earlier protagonists of today's multi-cloud era are Cloud Federations. Cloud federations brought the cloud computing to the next level, realizing much more than inter-cloud interoperability, rather providing a unified view of a heterogeneous pool of resources while using a single access point to control applications. In the Cloud Federation model, a number of cloud providers voluntary join their resources to collaboratively increase their market and to achieve scale economy that would have been outside their reach. Therefore, in cloud federation, an application is submitted via a specific cloud provider but its execution can in principle involve any combination of providers within the federation.

Thanks to the innovations in virtualization solution and approaches, the original federation concept has evolved over the time in more complex and functionality-rich paradigms. Among them, one of the most relevant is the *mobile cloud*, in which specialized devices at the edge of the network share the same execution environment of large cloud datacenter, in principle allowing a seamless migration of computation back and forth on both environments. The devices at the edge of the network participate in a collective realization of services, and they receive computation tasks and data to support demanding applications that would have been otherwise outside the computational capacities of the mobile devices.

BASMATI aims at delivering an integrated platform that will support the dynamic needs of mobile applications and users focusing on four main axes: (1) enablement of mobile cloud services, (2) federation of cloud infrastructures, (3) scalable infrastructure management, and (4) brokerage and offloading. This report provides a wide review of the most recent scientific and technological advancements in relation to the aforementioned axes, which will serve as the basis for the whole project.

1.1 Relationship to other Deliverables

This document provides an up-to-date review of the state-of-the-art literature that will serve as the foundation for the work performed within BASMATI. In particular, it sets the requirements for the global architecture design and specify the environment for the use case analysis.

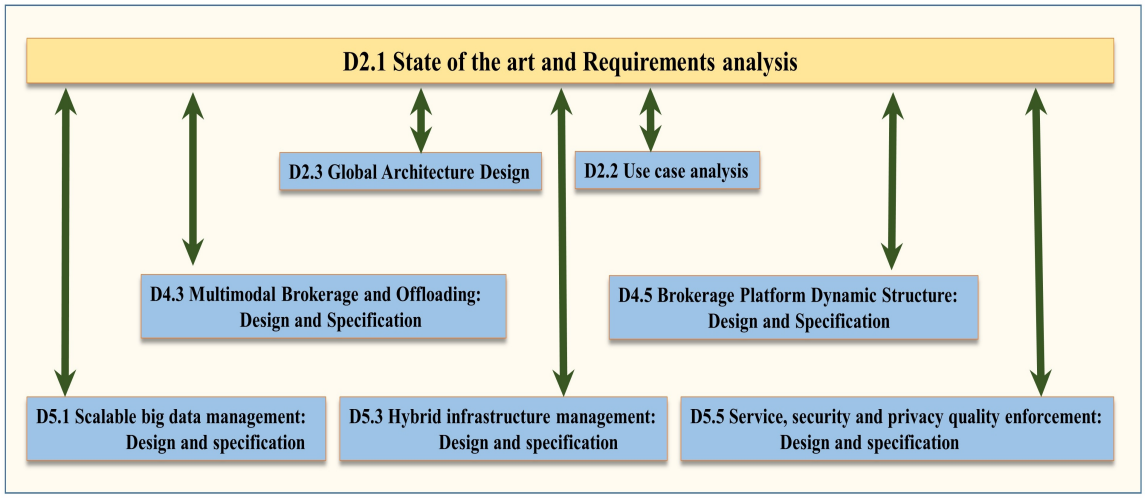

Figure 1. Relationship to other deliverables

In addition to this, the requirements and specifications identified in this deliverable will also determine the requirements that have to be considered for the brokerage platform, the service placement, the off-loading model, and the service handover. The deliverable provides details on the specifications for data management, hybrid infrastructure management, service monitoring, and QoS enforcement.

During the course of the project, the technologies and requirements related to BASMATI will continue being investigated (as aforementioned, this deliverable will be a live document, continuously updated even after its delivery), in order to ensure that the objectives and innovations of the project are valid, work is performed taking into account the latest state of the art and developments fulfill the identified goals and requirements.

1.2 Outline of Deliverable

The deliverable document is organized as follows. Section 2 of the document provides a comprehensive coverage of all the aspects related to mobile cloud, including the current architecture and technologies, methods used for user mobility analysis, service deployment, service monitoring, situational knowledge extraction and data management. Section 3 presents the concepts and baseline technologies of cloud federation. The economic models, Service Level Agreement, and network interoperability required to realize a cloud federation is also discussed. Section 4 deals with different types of service level agreements included in the BASMATI architecture. Section 5 presents cloud resource brokerage and the optimization methods. Each of those Sections presents topics the analysis of which is essential to realize the BASMATI vision. For each such topic, apart from the baseline technologies and the state of the art analysis, the reader will also find a first approach of the requirements analysis from a systemic perspective. Finally, Section 6, provides the results of the first requirements analysis at a use case level.

These sections and subsections are shown in following figure.

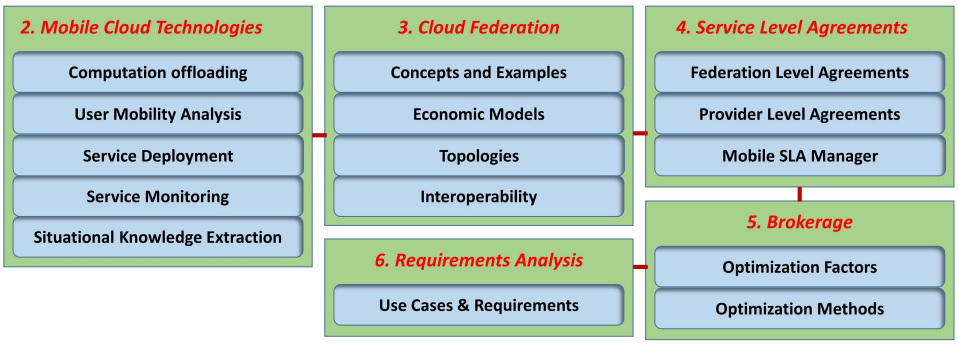

Figure 2. Outline of the deliverable

2 Mobile Cloud Technologies

This section presents state of the art on mobile cloud, service management frameworks, user mobility analysis and requirements to realize the objectives of BASMATI.

2.1 Architecture and Technologies

According to Othman (Othman et al., 2014), Mobile Cloud (MC) can be defined as "an integration of cloud computing technology with mobile devices to make the mobile devices resource-full in terms of computational power, memory, storage, energy, and context awareness". A core aspect of MC is computation offloading (also referred to as cyber foraging), that is the procedure to transfer resource intensive computations from a mobile device to a remote cloud computing server, with the aim of reducing battery consumption and augmenting the computational capacity of mobile devices. They differentiate between (i) infrastructure-based, and (ii) ad-hoc MC.

2.1.1 Infrastructure-based Mobile Cloud

In infrastructure-based MC, mobile devices act like thin client connecting through the mobile network to the remote server. In this case, the hardware infrastructure remains static and executes the application services and offloaded tasks. This model can also consider intermediate cloud resources, such as Cloudlet, whose concept was proposed by Satyanarayanan (Satyanarayanan et al., 2009) Cloudlet is intended as small clusters of relatively powerful computers in the proximity of the mobile devices and with connectivity to the remote cloud servers. Cloudlets are usually considered to be situated in common meeting areas such as concert areas, touristic city attractions, restaurants and coffee shops, such that mobile devices can connect and work as thin clients toward the cloudlet rather than to a remote cloud server (Fernando et al., 2012). Therefore, Cloudlets provide mobile devices with low latency and high bandwidth connections, thereby allowing an interactive response for demanding applications. The technologies at the core of MC and offloading are many and heterogeneous.

Several earlier solutions, such as Spectra (Flinn et al., 2002), employ Remote Procedure Call (RPC) to invoke remote computations. Despite their simplicity, the main disadvantages of these solutions are that the remote server needs to be equipped with all the code for execution beforehand, and developers need to write the application such that it can take advantage of the RPC invocations. Many other solutions exploit already existing models of applications in which there is a separation between the presentation of content and computation.

Cuckoo The Cuckoo framework (Kemp et al., 2010) is a framework that assists application developer in the creation of a mobile cloud application based on Java and targeting the Android operating system. Essentially, with Cuckoo it is possible to offload code to any running Java Virtual Machine, be it in a remote Cloud datacenter or in a cloudlet. The Cuckoo programming model exploits the difference between activities and services done by the Android operating system. Activities are single interaction operations with the user, while services are background tasks performing long-running operations or work for remote processes. Cuckoo identifies the latter as candidates for offloading, in fact supporting partial application offloading.

Calling the cloud Giurgiu (Giurgiu et al., 2009) proposes a middleware framework that supports dynamic partitioning of mobile applications to be offloaded to the cloud. The application is defined and orchestrated according to the OSGi standard (OSGi Alliance, 2007) based on Java. The mobile application is organized as a graph in which nodes and edges represent respectively "bundles" (i.e. modules in OSGi notation) and functional dependencies among bundles. The decision about what bundle to offload is performed by a graph cut, which is done according to some objective function. Other frameworks define their own concept of "offloading unit".

weblet Zhang et al. (2010) introduce the concept of weblet, which is defined as an independent component of a mobile application that can compute, store, and communicate in an agnostic way to its execution location (Zhang et al., 2010). Therefore, weblets can be executed both on cloud and on mobile devices. From a code perspective, weblets are interfaces in a high-level language (i.e. Java, C#) that have to be implemented by the application developer. When executing weblets in the cloud, Zhang et al. support replication, splitting and aggregation of weblets as means of optimizing the application. The offloading of weblets to the cloud is driven by a cost model, whose weights can be expressed by the mobile application developer.

µcloud Similarly, µCloud (March et al., 2011) proposes a model focusing on the composition of applications made of a set of heterogeneous components to support flexibility, reusability, and configurability. Components can be written once and used for multiple applications, even by different developers.

Several approaches rely on Virtual Machine (VM) migration to realize offloading. In this approach services run within VMs (or container of any flavours) that are migrated to the remote cloud server to realize offloading. VM-based offloading requires no or few modification to the code of the applications, but the drawback is that VM migration can be time and resource

consuming in such a degree that hides the benefit of offloading. The VM-based offloading also pairs seamlessly with the cloudlet model, which indeed provides small datacenter designed to accept VM as units of computation.

MAUI Cuervo et al. (2010) propose an energy-aware mobile application offloading framework. MAUI support fine-grained offloading to enhance energy saving with a negligible load on the programmer. It performs cost-benefit analysis by profiling each method in an application. MAUI exploits .NET as virtualization middleware and focuses on dynamic offloading at execution time, allowing the developer to annotate the part of the application to be offloaded.

CloneCloud Chun et al. (2011) propose an elastic execution framework that enables mobile devices to offload part of the application computation to the cloud. The framework employs dynamic profiling to collect data used in making the offloading decision, and static analysis to partition the application. The main goal of portioning is to optimize the overall execution cost. From an architectural point of view, CloneCloud creates clones of mobile devices in the remote cloud datacenters (or Cloudlet), which allow to seamlessly transferring computation from one to another. It does not require any programmer intervention (not even annotation) to perform offloading.

ThinkAir Kosta et al. (2012) propose ThinkAir, a dynamic and adaptive framework that supports on-demand resource allocation of mobile computation by simultaneously execute multiple offloading methods using VM images in the Cloud. VMs are dynamically created, resumed and destroyed upon necessity, with the ultimate objective to reduce mobile applications execution time and optimizing resource management in the Cloud. ThinkAir Framework consists of three main components: Programmer API and Compiler, Execution Controller and Application Server. The Programmer API permits the application developer to annotate methods of the application candidate to be off-loaded. With the provided annotations, the ThinkAir code generator, part of the compiler, produces removable method wrappers and included all necessary utility functions to enable these methods to be off-loaded. In order to support that generated code works both for ARM architectures in mobile devices and x86 architectures, the framework also provides a Customized Native Development Kit. Execution Controller takes into account current execution context and past executions information in order to decide whether to off-load a candidate method. The decision process is defined by four configurable policies, all policies aim to optimize a parameter or set of them and taking into account context information. These policies are: Execution time: based on historical data on about method's execution time; Energy: based on historical data on energy consumed; Execution time and Energy; Execution time and energy: optimization aims to optimize both parameters; Execution time, energy and cost: cost here related to incurred costs by executing in a public cloud infrastructure. Different three different profilers are used to support the off-loading decision: hardware, software and network profilers which inputs are used in conjunction to Energy Model. The hardware profiler collects CPU, Screen brightness levels, the power state of WIFI and 3G links. The software profiler collects per

each method either executed locally or remotely the following data: overall execution time, thread CPU time, number of executed instructions, number of calls, thread memory allocation size and Garbage collector invocation size while Network profiler collects information about Round Trip Time so to calculate perceived network bandwidth as well as other parameters from WIFI and 3G interfaces. Energy model in ThinkAir is enthused by PowerTutor model with modifications at the level of GPS and audio. The Application Server represents the cloud side of the framework. It manages Cloud resources considering both elasticity and scalability. Six types of VMs- VM templates- are considered. They differ in the number of CPUs allocated (from 1 to 8), memory size (200MB to 1024 MB) and Heap Size (32 MB to 100 MB). Basic VM type represents the smallest one, and it is allocated by default as the primary server. The primary server, it is a VM that clones the mobile device replicating both data and applications in order to fulfill specific user QoS expectations. AS an example, a user can have specific QoS requirements (e.g. completion time) for different tasks at different times, therefore the VM manager needs to dynamically allocate the number of VMs to achieve the user expectations. This primary server is always set-up ready to be contacted by the mobile device. Other VMs different to the primary server, called secondary servers, are instantiated on-demand by the user. The primary server manages communications from the mobile, the life-cycle of these secondary servers, as well as task, allocation in case of parallelization.

exCloud (Ma, Lam & Wang, 2011) provides a mechanism called stack-on-demand (SOD) that can perform fine-grained offloading of a Java Virtual Machine (JVM). By using SOD it is possible to avoid migrating the whole VM since SOD allows migrating of only the top stack frame of the execution, while the required code and heap data are brought in on demand subsequently.

Mobicloud (Xing et al., 2012) is another model in infrastructure-based MC that realizes geographic-based mobile cloud. In this model, mobile devices offload services to the cloud infrastructure that can be composed by multiple clusters around the world. Mobicloud analyzes the requests coming from the mobile and choose the correct server with a distributed scheduling algorithm based on the resource availability in the different clusters.

Phone2Cloud Tarkoma et al. (2014) have developed a computation offloading-based system for energy saving on smartphones called phone2cloud. The main aim of this system is to fully or partially offload the application from smartphones to the cloud to reduce the energy consumption, reduce the execution time, and improve user's experience, i.e. meet user's delay-tolerance threshold (Tarkoma et al., 2014).

Magurawalage et al. (2014) proposed system architecture for mobile cloud computing (MCC) that includes a cloudlet layer located between mobile devices and their cloud infrastructure or clones. This middle layer is called a cloudlet layer as it composed of cloudlets. Cloudlets are deployed next to IEEE 802.11 access points and serve as a localized service point closed to mobile devices to improve the mobile cloud services performance. They proposed an offloading

algorithm on top of this architecture with the purpose of deciding whether to offload to a clone or a cloudlet (Magurawalage et al., 2014).

2.1.2 Ad-hoc Mobile Cloud

The Ad-hoc Mobile Cloud approach organizes the collective resources of the various mobile devices in the local vicinity in order to create a virtual-cloud. In such context, a mobile device will use the resources of other close devices (the virtual cloud) instead of its own, in the same way, it would do with a remote datacenter. In principle, this approach can support high user mobility and create virtual Cloud on demand according to the necessity.

Several ad-hoc mobile cloud frameworks try to recreate typical cluster computation (such as MapReduce) in a virtual cloud composed of mobile devices. The approach presented by Huerta-Canepa and Lee (Huerta-Canepa and Lee, 2009) realizes a Hadoop (Shvachko et al., 2010) computation on top of a virtual cloud. Hyrax (Marinelli, 2009) supports a distributed computation based on Hadoop on a virtual cloud as well, also including the Hadoop Distributed File System (HDFS) for the storage.

Ghasemi-Falavarjani et al. (2015) developed a context-aware offloading middleware for mobile cloud (OMMC) to collect contextual information of mobile devices. By considering neighboring mobile devices as service providers, they investigated the resource allocation problem to select service providers that minimizes the completion time of the offloading along maximizing lifetime of mobile devices satisfying deadline constraint (Ghasemi-Falavarjani et al., 2015).

Pu et al. (2016) proposed a device-to-device (D2D) Fogging framework for mobile task offloading based on network-assisted D2D collaboration, where mobile users can dynamically and beneficially share the computation and communication resources among each other via the control assistance by the network operators. The purpose of their D2D Fogging is to achieve energy efficient task executions for network wide users (Pu et al., 2016).

2.2 User Mobility Analysis and Modeling

With the increasing usage of smartphone and positioning enabled devices a huge quantity of geo-localized data from moving objects like humans, animals or vehicles is being collected. These mobility data is represented as trajectories, namely the sequence of spatio-temporal coordinates of each position sampled by the device. These complex data come in huge amount and many techniques have been proposed in the literature in the last few years to represent, manage, analyze and mining such data (Renso et al., 2013a). The ultimate goal in mobility data processing is to solve high-level issues such as understanding how, when, where, and ultimately why objects move.

Elaborating the answer to these questions relies on a complex, multistep process, where the data sent by the data acquisition device (e.g., a GPS/GSM device) are analyzed and transformed to be gradually turned into something readily meaningful for the targeted application. This

process is sometimes referred to as Trajectory Knowledge Discovery process, Trajectory Data Mining, Mobility Data Mining.

When analyzing traces of human beings the privacy of the individual may be disclosed. A large part of the literature on mobility mining investigates methods to perform analysis while preserving the privacy of individuals (Renso et al., 2013). Privacy is a crucial issue to take into consideration when tracking individuals.

2.2.1 Analysis of Mobility Data

The kind of analysis that can be performed in large collections of trajectory data are limitless and mainly depends on the kind of moving object being tracked and the application needs.

The first class of methods introduced for studying mobility was inspired by physics, with the tentative model of the human movement with a physics law. A number of experiments have been done trying to capture a large amount of human mobility data, like the banknotes movements and later the phone calls in the US (Gonzalez et al., 2008). Authors found that humans tend to move following a truncated Levy-flight law. So, most people usually travel in close vicinity to their home locations, while a few frequently make long journeys. The most important result was the finding that all individuals seem to follow the same universal probability distribution that an individual is in a given position (x, y). Individuals display significant regularity, returning to a few highly frequented locations, such as home or work.

More recent works about finding the physical law to describe human mobility focus on the movement between a source and a destination location. Newton's law of universal gravitation has been used to derive the gravity model for mobility. This model assumes that the number of people moving from two locations is proportional to the populations of the source and destination locations and that it decreases with the distance (Simini et al., 2012a). A variant based on the simple particle diffusion model is the radiation model, where particles emitted at a given location have a certain probability of spreading into the surrounding areas. In this model the number of moving people from two locations is estimated depending on the origin population, the destination population (as in the gravity model), and the population in a circle whose center is the origin and radius is the distance between the origin and the destination, minus the population at the origin and the population at the destination. As discussed in (Simini et al., 2012a) and (Masucci et al.,2013), the gravity model does not account for fluctuations in the number of travelers between two locations and has many other limitations, for example, the parameter-dependency, the need for traffic data to fit the parameters, and the fact to be deterministic.

2.2.1.1 GPS Data

A different perspective is offered by methods that use mining to extract useful patterns form large amounts of GPS data. GPS data comes in a format that reports the spatial coordinates of a tracked object with a temporal timestamp. The sampling rate may vary depending on the

device, from few seconds to minutes or hours. The spatial accuracy of GPS is around 10 meters in outdoor space while it is usually not working or with a low accuracy in indoor spaces. GPS data are commonly used to monitor traffic as many vehicles are now equipped with GPS devices. A comprehensive example of trajectory mining performed on car trajectories for traffic application is the paper (Giannotti et al., 2011). Here authors report a comprehensive analysis of GPS data collected from cars moving in the city of Milan in Italy. This is a good example of which kinds of analysis can be done when large GPS car trajectories are available. From these GPS data, they measured some basic statistics describing the trips represented by the trajectories and a number of complex analytical questions explained later. One example of simple statistics includes the trip length and duration, from which we notice how mobility data show skewed distributions thus indicating a huge variability and heterogeneity of trips, which is typical of human movement. The correlation of length and speed of trips shows how only slow trips are very long, coherently with the intuition that many trips are slow because they take place in heavy traffic. The radius of gyration is another standard measure of movement that indicates how far an object moves from its preferred location (say home and work as an example) and, for each vehicle it can be computed as its average distance from the preferred location.

Another basic operation that can be done on mobility data is the Origin-Destination Matrix (OD matrix) very common in transportation research. This structure summarizes the flows of mobile users per temporal unit between two destinations and largely used in traffic applications.

Apart from the basic statistics, the most interesting and challenging analytical questions can be answered by data mining algorithms finding hidden mobility patterns. In the example reported in the paper (Giannotti et al., 2011) the analysis is aimed at discovering interesting subgroups of vehicles and travels characterized by some common movement behavior. Some typical questions are the following:

1. What are the most popular itineraries followed from the origin to the destination of people's travels? What routes, what timing, what volume for each such itinerary? More specifically, how do people leave the city center to reach the suburb area? What is the spatio-temporal distribution of such trips?

2. How to understand the accessibility to key mobility attractors, such as large facilities, railway or bus stations, main parking lots? How do people behave when approaching a crucial attractor, such as the airport short-term parking lot? What is the spatio-temporal distribution of such (a portion of) trips?

3. How to detect an extraordinary event – an important football match, the concert of a popular artist or an unusual traffic jam – and understand the associated mobility behavior? How and when do people reach and leave the event's location? What is the spatio-temporal distribution of such (a portion of) trips?

2.2.1.2 Trajectory Data Mining

These and other analytical questions can be answered thanks to the use of basic mining algorithms dealing with trajectories. The most common one is clustering where trajectories are grouped together by some measure of spatial and/or temporal similarity.

Clustering is also known as unsupervised classification since the objective is to find a way to put objects into groups without any prior knowledge of which groups might exist, and what their objects look like. Typical mining algorithms adapted or develop for trajectories are K-means, DBSCAN, T-OPTICS (Renso et al., 2013). This method is useful to find common groups of objects moving together, therefore detecting traffic jams or flows of vehicles following the same routes.

A predictive model is able to forecast the future locations that the object will visit. These models are usually built from the history of past behaviors. Route or trajectory prediction algorithms attempt to predict the path that a vehicle will follow in the future assuming its current position is known, but the vehicle's final destination is unknown (Pecher, Hunter & Fujimoto, 2016). These algorithms assume other information is available such as the trajectory taken by the vehicle thus far and/or the routes taken by other vehicles in its vicinity or derived from historical information. Several algorithms have been developed to predict future trajectories.

In literature, we found a pioneering approach called WhereNext (Monreale et al., 2009). WhereNext extracts sequential patterns called T-Patterns from a training data set of trajectories and combines them into a tree structure similar to a prefix-tree. In particular, each root-to-node path corresponds to a T-pattern, and root-to-leaf paths correspond to maximal patterns. When a new trajectory is presented, its most recent segment is compared against the regions represented in the tree, looking for the best match among the root-to-node paths. Then, the model finds that the matched sequence is a prefix of a longer pattern, and so it suggests a likely continuation region.

Another early work in destination prediction was developed in (Newson & Krumm, 2009). This work utilizes a Bayesian model that uses the immediate past trajectory taken by a vehicle to predict the vehicle's intended destination. An underlying assumption used in this work is that drivers utilize efficient routes in order to reach their intended destination. An efficient Markov model for destination and trajectory prediction is presented in (Pecher, Hunter & Fujimoto, 2014). When a prediction is requested, a data structure is traversed that holds the partial trajectory observed for the vehicle thus far. This data structure subsequently provides the empirical distribution of the forward trajectory. The previous five locations visited by the target vehicle are used to estimate the future trajectory by training a feedforward artificial neural network with two hidden layers consisting of 500 neurons each. In 2008, a Carnage Mellon University group published the PROCAB model (Probabilistic Reasoning from Observed Context-Aware Behavior) (Ziebart et al., 2008). This model uses the current context (e.g., accidents or congestion) and the user's preferences (e.g., fuel efficiency or safety) in order to predict his/her actions, rather than just focus on previous actions for prediction. The model maps actions into a

Markov Decision Process (MDP), where intersections are encoded as states and road segments are encoded as transitions. State transitions are associated with a cost, namely the sum-product of road segment features and cost weights obtained from collected training data. The model assumes that drivers attempt to minimize the cost to reach their destination. For destination prediction, it has been reported that PROCAB outperformed Newson & Krumm's Predestination algorithm for the first half of the trip. The PROCAB implementation used in (Ziebart et al., 2008) takes into account specific road features such as the number of lanes and speed limit.

In (Gui, Adjouadi & Rishe, 2009) the authors suggest using a personalization profile of smartphone users for localized searches. User activities and on-device sensors are queried to build a context profile, including demographical features and previous user activities. The weights of this information along with an environmental profile (weather, temperature, etc.) are trained via an artificial neural network. These profiles are used to rank personalized queries (e.g., local businesses). One could envision utilizing the same model to predict destinations or routes, similar to PROCAB.

In (Simmons et al.,2006) a hidden Markov model (HMM) is used to estimate a particular driver's destination and route, by using his/her previous trajectories along with driving time.

2.2.1.3 GSM data

GSM data as collected by the telecom companies are typical of two kinds: the handover representing the flow of users crossing cells, and Call Data Record (CDR) reporting the billing information regarding the calls made from the user. These data produce two different viewpoints of mobility of mobile phone carrying people. The handover data enable aggregations only over space and time. The main significant measure is the sum of the fluxes, which is calculated by fixing a space (a cell or groups of cells) and summing the fluxes over time (by hours, day or part of the day, and so on). This measure is used in order to get an idea of the collective movement patterns or to support the balancing tasks of the antennas' load. Given that, CDR data, explicitly refer to the user and they enable a richer set of measures to be computed over the three dimensions. The timestamp identifies the temporal dimension, the identifier of the cell where the call has started and the cell where the call ended provide the spatial dimension, while the caller id identifies the individual dimension. Given the three dimensions, it is possible to compute basic measures that depend on a single dimension, or on a combination of the three.

A basic measure is the number of users, which counts the distinct users in the dataset, how this number changes over the time can be extremely useful for studying the fluxes of people or the presence trend (Furletti et al., 2012). By aggregating data over two or more dimensions, new interesting measures can be identified. Considering the combination of space and time, we can count, for example, the number of calls for a given city during a given week or the temporal distance of calls in a given cell during a given month. Aggregating individuals over time, the distribution of presence in a zone or the number of distinct locations visited can be calculated. A

presence profile can be reconstructed starting from the call events for each user and aggregating them over space and time. In (Furletti et al., 2013) the authors describe the Individual Call Profile (ICP) as a matrix where each cell contains the number of presences of the individual in that particular time unit in the area under analysis. Of particular importance is the number of places visited over time that, in its simplest form, is estimated by counting the number of different antennas serving the user's calls in the time slot of interest. As explained in (Song et al., 2010), this measure can be used for identifying the frequency of returns to previously visited locations. This value increases, showing a decreasing tendency of individuals to visit previously unvisited locations. On the other hand, the probability of a user visiting a given location, the k^{th}-most visited location, follows the Zipf's law. Another complex measure derived by CDRs is the travel time. As reported in (Kujala, Aledavood & Saramäki, 2016), typical travel times in city-to-city trips can be computed by considering all the times between consecutive appearances in pairs of cities.

2.2.1.4 Predictive models of human mobility

Another class of methods used to analyze mobile phones is Markovian models, which describe the probability of a movement from an origin to a destination based on the recent history of the movements performed. In mobile data analysis, Markovian models (or Markovian chains, when using discrete sets of times) are generally used to predict a location based on the previous already known locations or sequences of cells. This case is modeled as an order-1 Markov chain, where the transition probabilities are dependent on individual user movement and represent the probabilities of moving from a single cell to one of its neighboring cells. In some papers, such as (De Mulder et al., 2008), the location profiles based on the model, are used to perform identification in a cellular network of a mobile user based on the location profiles of all users. Markovian models have also been used for general models about predictability, as in (Lu et al., 2013), where the authors find that the theoretical maximum predictability is 88% results proved by means of Markovian models. Their findings indicate that human mobility is highly dependent on historical behaviors and that the maximum predictability is not only a fundamental theoretical limit for potential predictive power but also an approachable target for prediction accuracy. Markovian models are useful when studying the mobility of humans at a daily scale, as in (Schneider et al., 2013). The framework they propose is based on Markov chain modeling periods of high-frequency trips followed by periods of low-frequency activity. A hidden Markov Model (HMM) is a special kind of Markovian model with unobserved (or hidden) states. HMMs are exploited for example in (Xu et al., 2011), where the authors aim to find the means of transportation being used by moving individuals. The hidden states are the transportation modes, while the observed states are the data collected by mobile phones.

2.2.1.5 Mobility and events in social media

Large popular events like musical and food meetings and similar are usually well reflected on Social media. The social media provide information about the users by their posts, generally a short text, tags and /or photos, links, video. Posts are sometimes accompanied by geolocation

thus providing the spatial and temporal coordinates of the user location at the moment of the post. Large events and social media have been studied in a number of papers in the literature. The main categories of study that we identified are (1) recommending events to users (Quercia et al., 2014; Macedo, Marinho & Santos, 2015), (2) studying factors affecting the users' participation in events (Mascolo, Noulas & Mascolo, 2014) and (3) estimating the number of attendees in a given event (Botta, Moat & Preis, 2015).

Within the first category, event recommendation, Quercia et al. (Quercia et al., 2014) proposed a recommendation approach to suggest events to users based on mobile phone data usage. Similarly, Macedo et al. in (Macedo, Marinho & Santos, 2015) and Wang et al. in (Wang et al., 2016) addressed the challenge of recommending events within event-based social networks (EBSNs). Each of these approaches is challenged by the cold-start problem, in that a history of the user's previous event attendances may not be available, and recommendation evidence may resort to the events that are geographically closest (Quercia et al., 2014).

The study by Georgiev & Mascolo (2014) is an example of a contribution in the second category. In particular, they addressed the extent to which geospatial, temporal, and social factors influence the users' preferences towards events (Georgiev & Mascolo, 2014). They formulated a predictive modeling task trying to match a user's mobility profile against the collective past Foursquare check-in activity of potential event attendees. They took into account the homophily effects on the users' event choices as reflected by location-based social media.

Finally, within the third category of related works, Botta et al. in (Botta, Moat & Preis, 2015) investigated whether mobile phone usage and the geolocated Twitter data can be used to estimate the number of people in a specific area at a given time. In considering two case studies of access-restricted areas in Italy: a stadium and an airport (where they had access to ground true visitor numbers), they concluded that geolocated tweets with mobile phone data could be a good proxy of estimating the number of users.

In (Cesario et al., 2016), the authors described a methodology for identifying the user behavior and mobility patterns of Instagram social network users visiting the EXPO 2015 world fair in Milan, Italy. They were able to discover how the number of visitors changed over time, identify the most frequent sets of visited pavilions, which countries the visitors came from, and the main destinations of foreign visitors to Italian regions and cities after their visit to the EXPO. A similar work by the same authors (Cesario et al., 2015) analyzed geotagged tweets of people attending the 2014 FIFA World Cup. In doing so, they identified the most frequent movements of fans, the number of matches attended by groups of fans, clusters of most attended matches and the most frequented stadiums.

2.2.1.6 Indoor Mobility

An increasing interest is receiving the study of mobility of people in indoor spaces like museums, shopping malls, hospitals and large events. The characteristics of these environments is that GPS

is typically not working in indoor spaces while the GSM granularity is too coarse (one indoor space is typically covered by only one cell). This is why these environments need ad hoc positioning systems to detect the movements of inside people.

The most common are RFID, Wifi, BluetoothLE (Bluetooth Low Energy). All these methods provide complementary features, but usually, they detect only the proximity and using triangulation techniques the users' position accuracy may improve, but none of them may accurately detect the individual movements with a high accuracy as GPS, thus they cannot produce similar accurate trajectories. Mobility analysis, therefore, has to deal with this imprecision.

An example of usage of indoor positioning systems (namely BluetoothLE) at Louvre Museum describing an interesting analysis of the visitors' movements is reported in the paper (Yoshimura et al., 2014). Here authors study how users visit the museum by answering the following analysis questions: Do visitors always enter from the same lobby? Do longer stay implies longer visits? Which is the order in which the art works are visited? What do the short visit users visit most? Are they avoiding crowded areas? Where do they stay longer, looking at which artwork?

User Mobility analysis is one special case of analysis of context information. In order to ensure a broad usefulness of the envisioned solution, we cope with the analysis of context information in general thereby addressing not only mobility based on plain geo locations (GPS) but also based on network information and even time and time intervals or information about the device used.

The following citation gives a useful definition and perspective of what we deal with: "Such context-aware systems adapt according to the location of use, the collection of nearby people, hosts, and accessible devices. A system with these capabilities can examine the computing environment and react to changes to the environment" (Schilit, Adams & Want, 1994).

Smartphone today potentially provide a lot of different context information via so-called context sensors as depicted in Figure 3: Context sensors of a smartphoneFigure . In general, any sensor information could be analyzed here as long as they give insights about the user´s current situation.

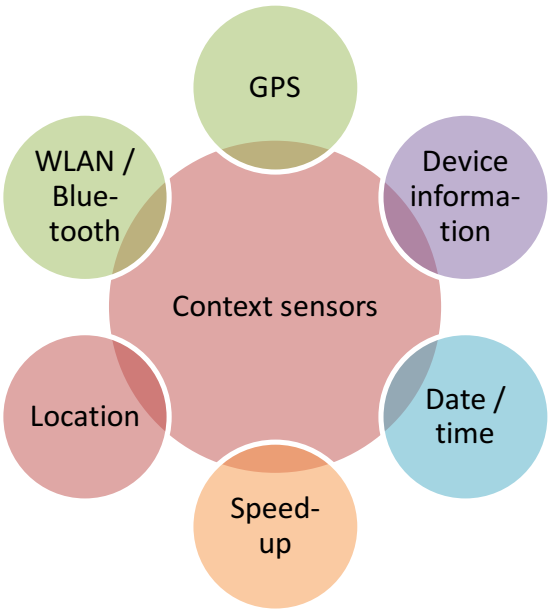

Figure 3: Context sensors of a smartphone

The following section will describe the state of the art of context frameworks. The selection of frameworks described here is mainly based on the following assumptions:

- Context information should be collected and sent to (mobile) clients.
- Context information should be analyzed on a central server.
- The technologies used by the framework should be easy to be integrated into the BASMATI use case demonstrators and the BASMATI components.

2.2.2 Context Frameworks

2.2.2.1 AWARE

Developed from 2013 on until today, by the University of Oulu (Finland), the AWARE framework might be seen as one of the most promising context frameworks. It consists of a server and clients for Android, iOS and OSX Desktop. Each client provides 28 context sensors collecting and delivering contextual information to a central server via HTTP-Post. The set of sensors to be used can be configured. The server stores the information in a MySQL database as depicted in Figure . On the server side, the collected information can be grouped and protected from each other via a concept analogous to multi-tenancy (so-called studies in AWARE terminology).

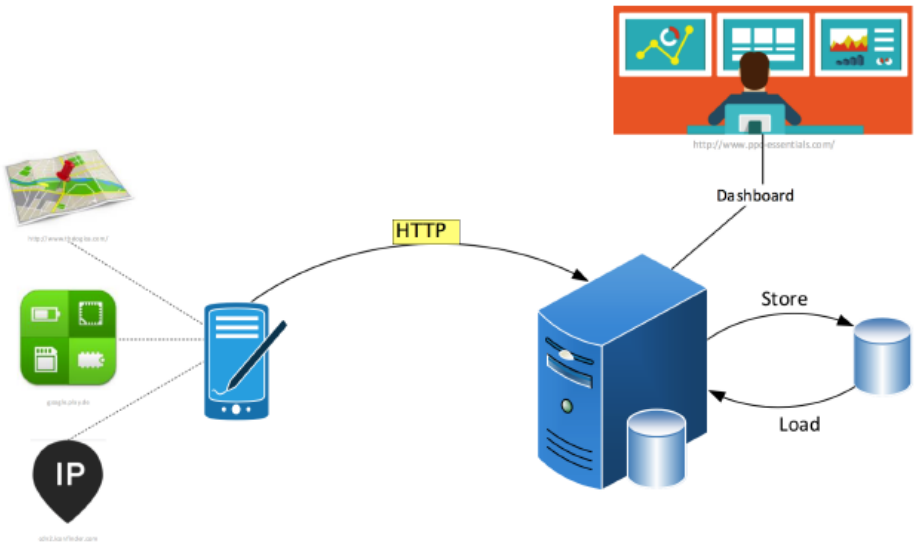

Figure 4: Architecture overview of the AWARE framework [Source: Schork, 2016]

The 28 context sensors available in the AWARE clients are able to collect the following exemplary

- Position
  - GPS
  - Sensorfusion
- Network
  - GSM
  - WLAN/Bluetooth
- Physical Sensors
  - Speed-Up
  - Location
  - Brightness
- Device Information
  - Battery
  - Operating system
  - Applications
- Komplexe Sensoren
  - (Google) Activity
  - Surveys

2.2.2.2 CTK – The Context Toolkit

Developed from 2000 until 2013, the CTK (Context Toolkit) from the University Berkeley is based on the assumption that the context is the context of an application (not the user) and that the

application itself can collect the relevant contextual information. The CTK has the following characteristics:

- Encapsulation of sensors
- Access to context information via network API
- Abstraction of context information via interpreter
- Sharing of context information via distributed infrastructure
- Storage of context information and their changes over time
- Access control in case of private/sensible information

The framework provides relevant conceptual work and insights; software applications based on CTK are rather limited (Nagel, 2001; University Berkeley, 2016). Implementations of clients, for example, are only available on P2P basis and not for diverse operating systems like android or iOS.

Conceptually, CTK is divided into the following components:

- BaseObject: Basic Peer to Peer (P2P) communication infrastructure and access point for applications
- Widget/Service: Call-backs to be called on context changes can be registered at the widgets (via subscriber model), see Figure 4.
- Aggregator: Collection of several widgets
- Interpreter: Mapping and fusion of context information (e.g. geo-coordinates will be mapped to a concreate city, two sensor data like position and speed-up will be mapped to a new activity.
- Discoverer: Finds new, available context components inside the P2P-Infrastructure

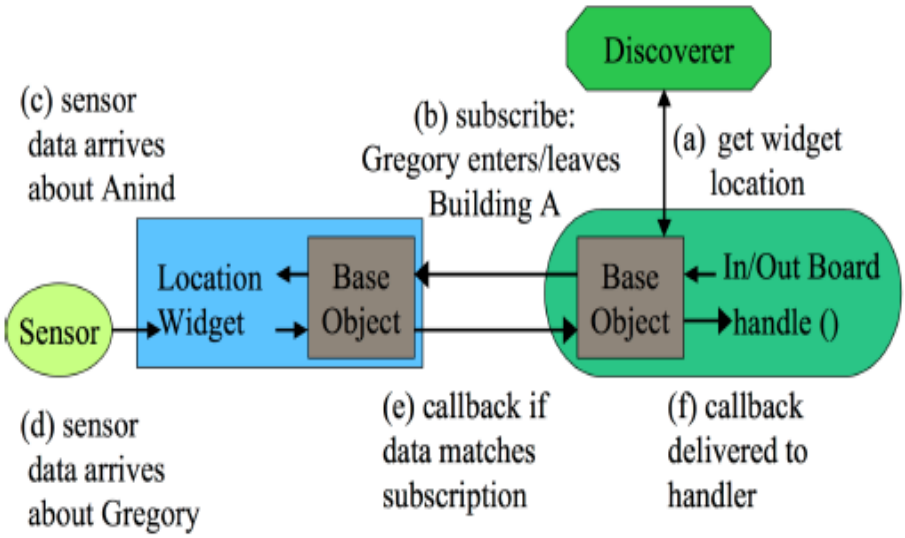

Figure 5: Call-back mechanism of CTK [Source: Dey, 2000]

Because of missing implementations in the clients and in the server, the CTK is not of highest relevance for the concrete usage within the BASMATI project even though it gives interesting conceptual insights.

2.2.2.3 JCAF – Java Context Awareness Framework

As CTK, JCAF (Java Context Awareness Framework) from the University of Aarhus (2003-2005) is based on a P2P communication model. The basic assumption of JCAF is to provide Java libraries to be consumed in concrete context-aware applications; it's based on Java RMI, serialization and dynamic loading and execution of code during runtime.

Basic concepts of JCAF are:

- Context services: Management of context information from entities (like retrieve, store, distribute context information)
- Entity: Describe Context information as tuple of the form (Entity, association, object), e.g. (User, uses, PC)
- Context Clients: Retrieve Context information; register either as "Context Monitor" or "context actuator" at a context service.
- Context Events: Occurs in relation to context service and context client

As depicted in Figure 5, context information is translated and aggregated within the Transformer Repository by either Translators or Aggregators. Access Control based thereon can be realized based on HTTP basic authentication.

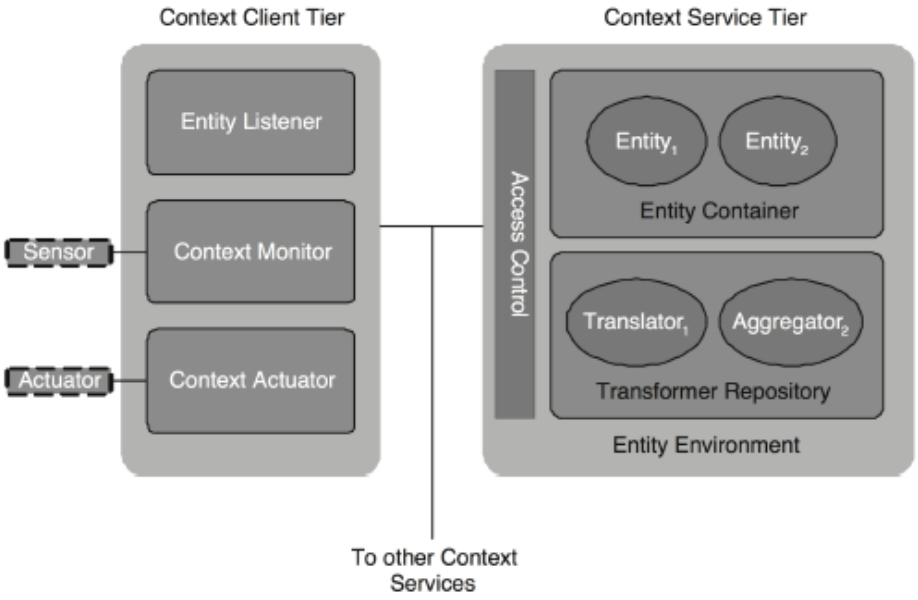

Figure 6: Architecture overview of JCAF [Source: Bardam, 2005]

Because of missing implementations of (mobile) clients and missing further development (last change 2009), the JCAF does not have the highest priority for use in BASMATI.

2.2.2.4 Hubiquitus

Hubiquitus, developed from 2012 until today, forms a messaging middleware based on JavaScript together with binding for Java, NodeJS and .NET. It supports datagram-based message transport and can be used on mobile platforms as well.

Hubiquitus is based on the following concepts:

- Actor: Identified by an "Actor-ID", send-method with arguments (target Actor-ID, content, timeout and Callback). The Actor-ID is composed of "Bare-ID" and Resource (like Domain/Resource in the REST terminology)
- Container: Group of Actors. For each Node-Process (cf. JavaScript Node.js) one container is allowed).
- Discovery: Resolves target Actor-IDs for P2P-Communikation between Actors by Inter-Cloud Service Management

Because of missing details and documentation and because Hubiquitus´ use is only mentioned without concreate reference, we´ll not consider it further for the use within BASMATI, even though it´s basic concepts and the approach seems promising.

The AWARE framework impresses through the comprehensive code base and ready-to-use implementation compared to the rather conceptual work provided by CTK, JCAF and Hubiquitus.

2.2.3 BASMATI User Mobility Analysis Requirements and Specification

The BASMATI project focuses on two different scenarios: (1) analyze the movements towards and inside a large event and (2) predict/detect the users who cross the national border. There are no specific approaches in the literature for these two tasks, therefore our aim is to adapt the existing technologies available for other similar applications to this specific case. We want to (1) exploit both GPS (when available) and social media data collected before, during and after a large event to understand the mobility of users and (2) to identify and predict from social media the cross-border cases.

In the first case the users can be tracked inside an event by collecting data through some ad-hoc app tracing the user movement by using GPS or by using ad-hoc positioning device installed at the event like the Bluetooth Low Energy (i.e. iBeacons) or Wifi networks and from these data we can understand the mobility patterns (flows of people, regularity of patterns, specific behavior, interests, etc) of users equipped with a smartphone (with Bluetooth or Wifi).

These two options have complementary benefits and drawbacks. GPS enabled *app* requires the users to install the *app* and, give their consent to use those data for the underlying purposes. That might raise privacy concerns. Moreover, they need to accept the high consumption of their

smartphone battery by the GPS. A tradeoff regarding battery consumption can be found by decreasing the sampling rate at the cost of losing accuracy. Indoor positioning techniques to monitor an event requires a higher investment in terms of cabling the environment with iBeacons or Wifi access point, but they do not require the use to install any software or do any action. They rely on the fact the users may have the Bluetooth on and/or be connected to the Wifi network. These techniques loose in position accuracy compare to GPS app, but they increase the potential number of tracked people.

A complementary approach is to collect social media posts about the event. For example, we can also predict the user's participation to an event by analyzing their posts to social media.

Social media can also be exploited in the second case to predict the users who are likely to cross the border (prediction) and the users who actually crossed the border in the past. We can take advantage of the geolocated media posts but also (due to their low number) the content of the post may help in detecting the user change of country.

In general, we can distinguish between different kinds of user or user mobility data relevant within the BASMATI use cases:

- Personal data
- Environmental data
- Real-time trajectory data
- Statistical data

The following examples for these kinds of data should help to understand what data we´ll need to implement the use cases. Together with the description of analysis approaches for mobility data, this forms the basis for defining parts of the BASMATI architecture and the concepts for Modelling Users and Applications and for the big data management.[1]

2.2.3.1 Personal Data

- Number of persons when moving in a group
- Special kinds of group members e.g. (small) children
- Travel information (means of travel, schedule)

[1] Cf. upcoming deliverables: D2.3 Global Architecture Design, D3.1 Analysis and Modelling of Users and Applications: Design and Specification Situational Knowledge Acquisition and Inter-Cloud Service Monitoring and D5.1 Scalable big data management: Design and specification.

- Configuration of user attributes, like personal interests and preferences regarding food, music, sports and similar as well as specific handicaps that might exclude certain POIs or paths
- Pre-selected POIs, like chosen food stand, concert/event, meeting point.
- Potentially static (allergies, blood group, handicaps etc.) and sensory health data knowledge in case of emergency [optional]

2.2.3.2 Environmental Data

- Weather information
- Network information like Wi-Fi/WLAN access points.
- Traffic reports / parking information
- Characteristics of the terrain (topology, paths, fences, entries)

2.2.3.3 Real-time Trajectory Data

- Occupancy of the terrain near a POI
- Queue at the entrance
- The length of the queue and calculated waiting time.
- GPS position every 30 seconds
- Total number of visitors on the area
- Occupancy of potential meeting spots/POIs
- Density of the crowd in front of and near the POI
- Any narrow places on the way to the next POI
- Density of crowds
- Movement speeds of peoples and crowds
- Distribution of crowds
- Number of persons on the area
- Geographical positions of all people

2.2.3.4 Statistical Data

- How many hours before the event is the place occupied?
- How many hours before the event are all the good seats in front of the stage occupied?
- Particularly important are characteristics of the terrain to avoid crowds of people and confusing places.
- Characteristics of the terrain, like narrow places and depressions.

2.3 Service Deployment: VM and Micro Services

There are various open source software projects to support service deployment on multi-cloud environment. We introduce some representative projects in this section.

2.3.1 ManageIQ

ManageIQ is a multi-cloud management platform software. ManageIQ is an open source version of Red Hat CloudForms technology, which enables admins to manage public clouds, private clouds, and virtual infrastructures in a single interface.

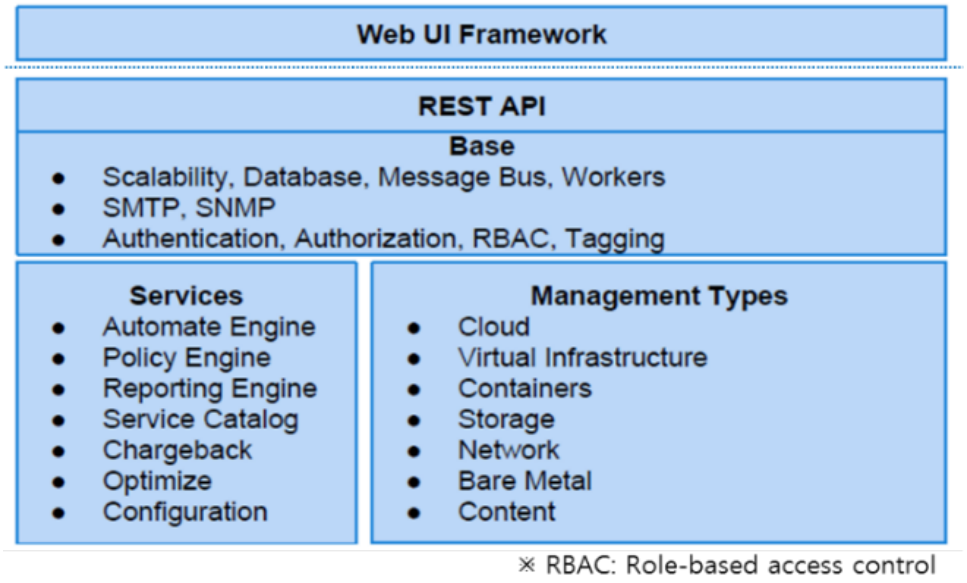

Figure 7. Structure of ManageIQ [Source: Neary, D. & Mark, J., 2016]

MangeIQ provides the following management features (Neary & Mark, 2016). Inventory management via smart state analysis, Self-service provisioning & service catalog, Capacity and utilization, Quotas and showback/chargeback, Configuration and change management, Policy engine and management, Automation and orchestration, and Reporting. ManageIQ can create auto-scalable cloud applications.

The user of ManageIQ should be a service provider. The user needs to develop a software stack for a service provisioning. Eventually, the user (a service provider) needs to establish SLA too.

2.3.2 Scalr

Scalr is a web-based open source cloud management platform, and the goal of Scalr is to make the managing and administering multi-cloud infrastructure and resources in the cloud. Scalr basically allows users to manage cloud resources provided by cloud service providers such as AWS, Rackspace, and etc. It also enables users to create a cloud environment where they can operate easily and quickly within the scope of a defined policy.

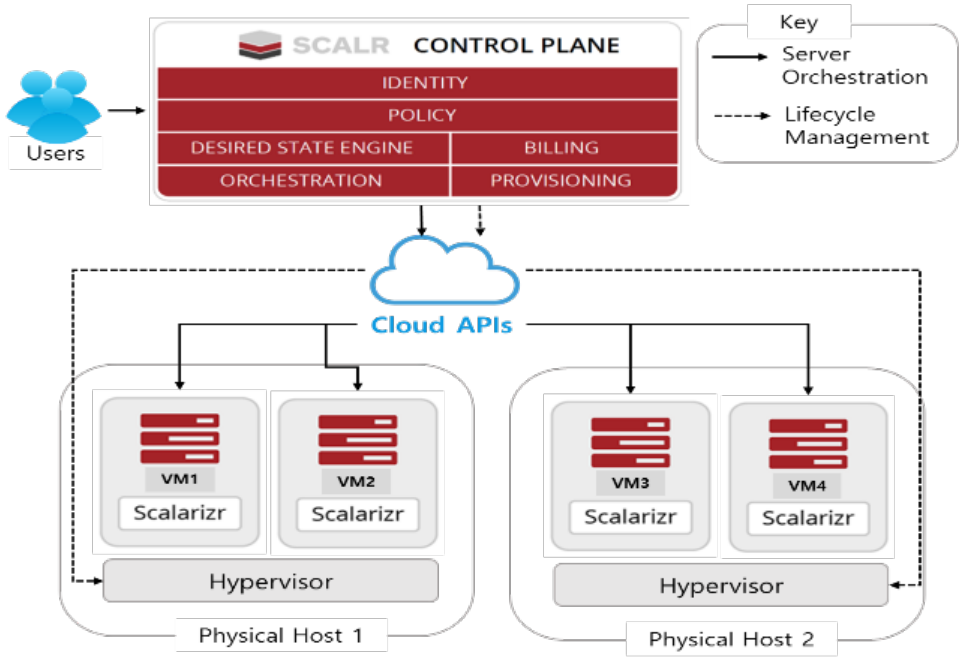

Figure 8. Structure of Scalr [Source: Harnik, 2016]

Scalr's key functionalities are as follows (SCARL, 2016). The hybrid cloud management makes users have a common set of access controls, policies and management practices. The cross-cloud API controls multi-cloud in a single interface. The orchestration engine automates handling event-based, manual, or scheduled state changes to the cloud infrastructure. The cloud scaling automates scaling of cloud resources based on CPU utilization or other manual demands.

However, Scalr does not provide a SLA management functionality. The user needs to develop a software stack for a service provisioning, and the user (a service provider) needs to establish SLA.

2.3.3 Stratos

Stratos (apache stratos, 2016) is an apache open source that is a highly-extensible platform-as-a-Service (PaaS) framework. Apache Stratos supports to run Apache Tomcat, PHP, and MySQL applications on cloud infrastructure.

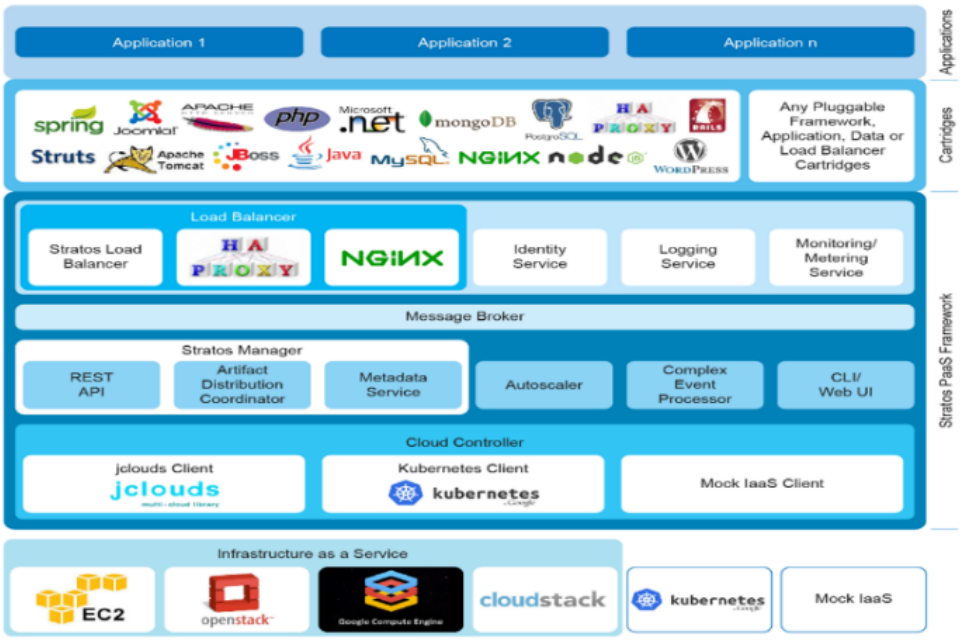

Figure 9. Architecture of Stratos [Source: apache Stratos, 2015]

Figure 8 shows Stratos architecture. Stratos has a multi-layered architecture, infrastructure as a service (IaaS), core components, cartridges, and applications. IaaS providing virtualized computing resources is the bottom layer in Stratos. Using jclouds (a library including a unified API for multi-cloud) (apache jclouds, 2016), Stratos can be deployed on IaaS, e.g. EC2, OpenStack, CloudStack, Google cloud. In addition, Stratos provides Docker with Google Kubernetes and Mock IaaS simulating IaaS. The core component of Stratos consists of Stratos Manager, auto-scaler, complex event processor (CEP), Load Balancer, Message Broker, identity/monitoring/metering services, and CLI/Web UI. Stratos Manager provides REST APIs to provision applications, and also monitor and meter the PaaS. Autoscaler is responsible for the elasticity of application and load balancer manages the traffic of load in Stratos. Stratos introduces a new concept called the cartridge that is a virtual machine with software components to interact with the Stratos. Stratos provides the following types of cartridges. Users in Stratos can execute various applications based on the cartridges such as spring, Tomcat, MySQL, and PHP on cloud environments.

2.3.4 Project Jellyfish

Project Jellyfish is an open source broker system offering policy-driven solutions that assist organizations with the management of multi-cloud IT environments (Project Jellyfish, 2016). Because it is not just a cloud management tool, but also an advanced business analytics and intelligence tool, it helps service providers to unify their cloud-based resources, infrastructure, and services through a centralized e-commerce platform and to satisfy organization's IT needs for cloud services using an easy to use self-service portal within budget.

The key functionalities of Project Jellyfish include providing search-and-compare wizard to evaluate various cloud services and cost across the multiple cloud infrastructure, user-friendly dashboard to monitor usage and cost data, allowing users to create projects with the defined services offered based on policy and workflow, monitoring and orchestration of resources, and automation of entire lifecycle of a cloud service.

Since Project Jellyfish is a CSB platform, CSB needs to establish SLA(CSP A, CSB), …, SLA(CSP K, CSB), and SLA(CSB, CSC). However, since Project Jellyfish does not provide value-added (reproduced) services, the agreement in SLA(CSB, CSC) is implicitly and identically defined by SLA(CSP, CSB).

2.3.5 CompatibleOne

CompatibleOne is an open source cloud service brokerage platform that provides IaaS brokerage from heterogeneous clouds. CompatibleOne is focused on the adoption of open standards to foster an open cloud computing ecosystem. For instance, CompatibleOne adopted OCCI (Open Cloud Computing Interface) standard (OCCI & CompatibleOne, 2012), which defines a meta-model for cloud resources and a RESTful protocol among the cloud resources, so that CompatibleOne can provide a strong interoperability and a high degree of extensibility of the platform. Along with OCCI, CompatibleOne platform consists of many micro services and they can communicate each other by using an object-based description model and a RESTful protocol to perform a service brokerage or management. CompatibleOne provides an object-based description model (CompatibleOne resource description model, CORDS) of cloud resources. Based on the description model, a user of CompatibleOne can make a manifest in XML to request or control a cloud service. The parser of CompatibleOne is responsible for parsing of the manifest and validating XML syntax so that micro services in CompatibleOne platform understand the requests from the user.

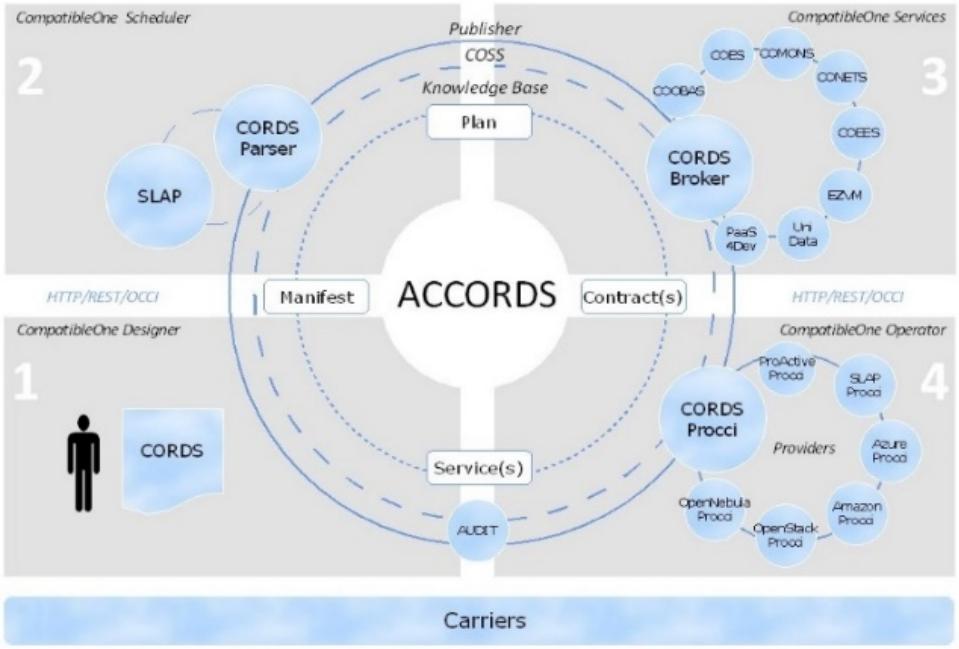

Figure 10. Architecture of CompatibleOne [Source: ComatibleOne, n.d]

Figure 9 shows the architecture of CompatibleOne. There are various micro services in the platform. For instance, SLAM (Service Level Agreement Manager), Parser which is responsible for parsing and validating XML manifests, COSS (CompatibleOne Security Service) which provides a Transport Layer Security (TLS) to secure all micro services and users. COMONS (CompatibleOne MONitoring Service) to manage monitoring of provisioned services, COPS (CompatibleOne Placement Service) to determine the optimal service placement (to which cloud), and COOBAS manages ordering, billing, and accounting.

2.3.6 BASMATI Service Deployment Requirements and Specifications

The service deployment feature performs the sequence of service provisioning and configuration actions that are generic across heterogeneous providers to achieve the objectives of the service lifecycle workflow (linked with service adaptation). For instance, if the lifecycle workflow requests to scale out, the service deployment will perform all the necessary steps to add a new server instance (launching a new VM, configuring the service, reconfiguring the load balancer…).

The requirement of service deployment is as below.

- Application and user context information to deploy service on best-fit cloud infrastructure based on user location, resource requirement, security level, cost, application workload and so on.

- The hybrid cloud management to makes users have a common set of access controls, policies and management practices. The cross-cloud API controls multi-cloud in a single interface.
- The orchestration automation to handle event-based, manual, or scheduled state changes to the cloud infrastructure. The cloud scaling automates scaling of cloud resources based on CPU utilization or other manual demands.
- REST APIs to provision applications, and also monitor and meter the service.
- Search-and-compare wizard to evaluate various cloud services and cost across the multiple cloud infrastructure.
- User-friendly dashboard to deploy the required specific service and monitor usage and cost data
- Automation of entire lifecycle of a deployed service.
- Object-based description model of cloud service. Based on the description model, a user can make a description to manage a specific cloud service.
- Automated Service Level Agreement (SLA). Most commercial cloud offerings rely on a model similar to end-user license agreements commonly found with software packages, which are based on the principle that a customer can either accept the provider's terms and use the offering, or disagree with them, which in consequence means he cannot use the offering. This model can be extended in a way to be a bit more dynamic and user-friendly by offering several levels of service for different prices, such as different levels of reliability or computing power. However, these models are still a far step away from the automated negotiation of SLA terms.

2.4 Service Monitoring

This section provides some information about technologies used for service monitoring in the cloud.

2.4.1 Federation Monitoring

2.4.1.1 Zabbix

The EU BonFIRE Multi-Cloud Test Facility (BONFIRE, 2014) offers a multi-site cloud testbed that supports large-scale testing of applications, services, and systems over multiple, geographically distributed, heterogeneous cloud testbeds.

The monitoring functionality of the BonFIRE is implemented based on a server-agent model. The server is deployed as a separate resource and collects monitoring data reported by the agents that reside in the VM images. To implement monitoring, BonFIRE has adopted the open source monitoring software Zabbix (ZABBIX, 2001).

Zabbix comprises two major software components: Zabbix server and Zabbix agent. The server is referred to as an"aggregator" in the introduced monitoring system (in BonFIRE as well). BonFIRE uses a special type of agent, the active Zabbix agent, in order to overcome possible accessibility

problems because of NAT. In this case, the agent is the one that initiates the communication to the server and sends the monitoring data. Agents are small software components configured to send metric values to the server at regular intervals. Agents typically produce metric values by executing Unix scripts written to obtain the value. The monitoring aggregator provides both a GUI to observe the monitoring metrics and also an API to support programmatic access to monitoring data.

2.4.1.2 RESERVOIR Federated Monitoring

When Service Clouds are federated to accept each other's workload there needs to be a consideration of how monitoring will behave in the presence of the federated Virtual Execution Environments (VEEs). Monitoring needs to continue to work correctly and reliably when a service executes across federated Clouds. When some VEEs are migrated to another Cloud, the monitoring data distribution mechanism will need to cross Cloud boundaries. It is essential that the interfaces and formats between Clouds be standardized in order that federation monitoring to work in heterogeneous environments. This will ensure that that the monitoring data for all the VEEs of a service will be connected, whether locally or remotely.

A different solution has to be taken for infrastructural monitoring information. One group of measurements (i.e., messages from VEEH Probes) are not intended to leave the Cloud managing each VEEH. In fact, monitoring information on the physical machines of a specific Cloud is needed for the Cloud management (e.g., placement, anomaly detection) and generally kept hidden from the outside. Data from Hypervisor VEE Probes, on the other hand, needs to be received both by the local Cloud Service Manager (e.g., for interCloud billing) as well as by the Service Manager at VEE originating site (e.g., for Service Provider billing).

The monitoring data plane has a per-service segment between federated sites, in order to transmit the monitoring data. This connection between monitoring data plane parts on each Cloud is shown as the Per-Service Segment in Figure 10.

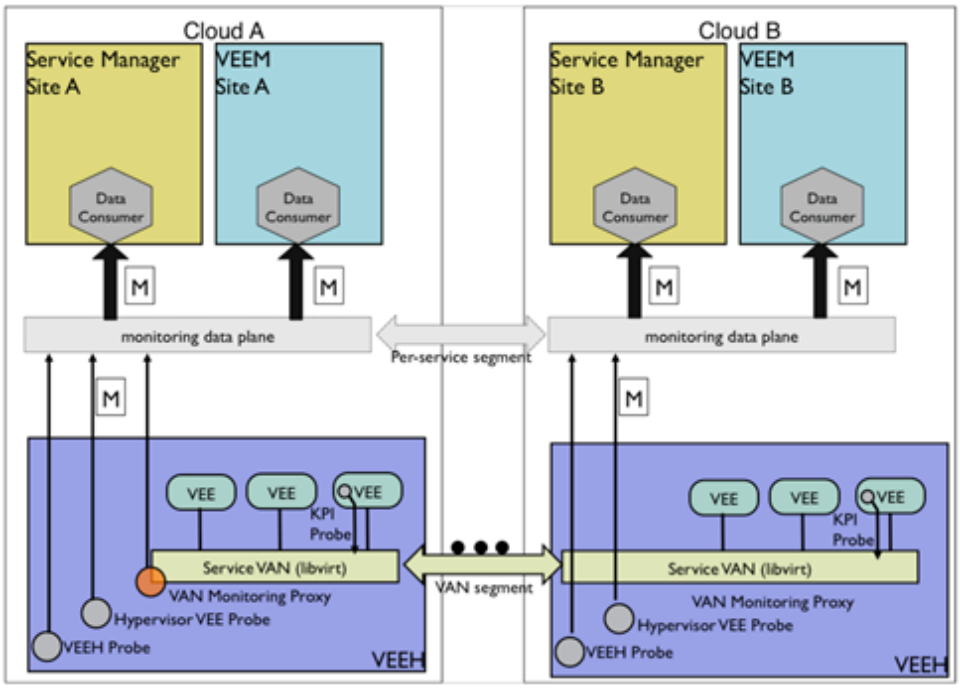

Figure 11: Monitoring across federated RESERVOIR Clouds [Source : Clayman, 2012]

When the first VEE of a service is about to be migrated across sites, a Federation Manager instantiates gateways for each service. A service gateway connects to the internal monitoring data plane of a RESERVOIR Cloud and forwards the relevant information to a matching gateway on the destination Cloud.

In Figure 11 we see 3 such service gateways, for service A, service B, and service C. Only monitoring data from service A VEEs crosses the service segment through the service A gateways. The same process applies for service B and service C.

The service gateways are under the control of the Federation Manager, which itself is controlled and managed by the VEEM. As service gateways are under administrative control of the Cloud that instantiates them, this enables:

- Complete control over the information that is chosen to be forwarded over the gateway, as the VEEM and Federation Manager can adjust the settings of the gateway;
- Gateways to be adaptable and implemented according to each Cloud monitoring data plane technology.

This allows for a Cloud to use one technology internally for the monitoring data plane, and use a different technology for the Cloud-to-Cloud transmission (e.g., multicast inside the Cloud and JMS over the gateway). Furthermore, two Clouds may have entirely different technologies for their monitoring data plane, but the gateways will provide the connection between them.

When VEEs are undeployed or migrated back to their originating site, and there are no more VEEs for a service still running on a Cloud, the federation managers on each Cloud have the responsibility of tearing down the appropriate service gateways.

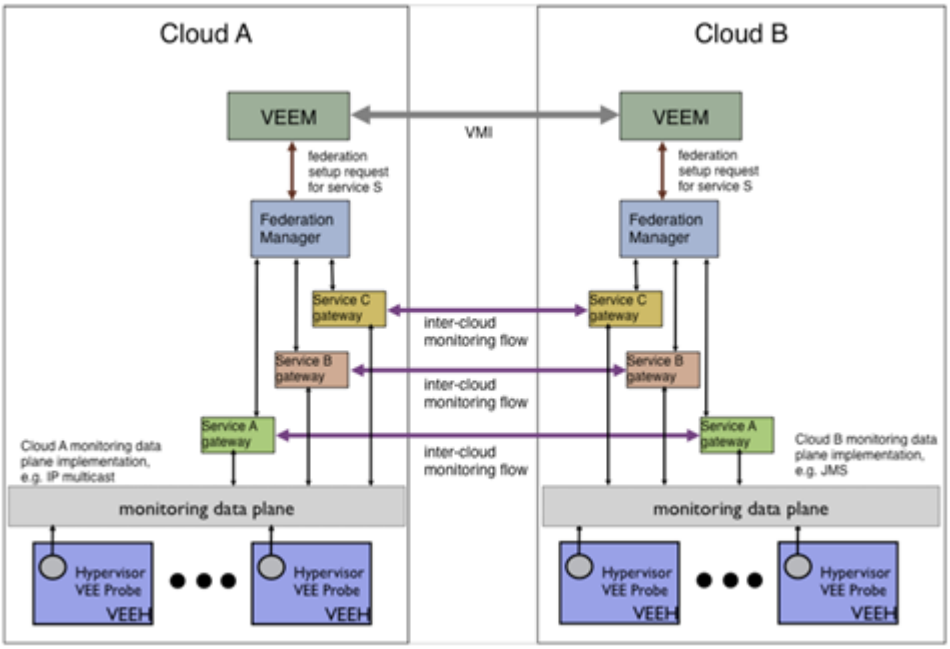

Figure 12: Federated monitoring management [Source: Clayman, 2012]

2.4.2 Resource Monitoring

The Resource monitoring is responsible for measuring the Key Performance Indicators (KPIs) of the systems and services. In cloud systems, it provides the data primarily for the following areas: a) system monitoring b) Accounting, billing, and auditing c) Service Level Agreements (SLAs). In system monitoring, it helps to diagnose hardware and software problems, to enhance the resource utilization and to ensure the system's performance and security. It also plays a key role for measuring services and for precisely charging the users based on their resources and services consumed. Sensors provide data for the monitoring component, and this includes information such as resource consumption of hardware and software and the Quality of Service (QoS). It is then forwarded to higher layers using web services or displayed to administrators using a GUI. The section below concentrates on resource monitoring only at the infrastructure level.

In general, the cloud software has its own monitoring component. Alternatively, external monitoring tools, such as Nagios (Nagios, 2009) and collectd (collectd, n.d) can be used using which monitoring information about the hardware and the services of the cloud software can be obtained from its respective plug-ins. Apart from the above data, cloud software also provides information on the total number of users, projects / tenants / accounts and the association between users / projects and virtual machines.

Cloud management systems such as OpenStack use VMM (libvirt) (libvirt project, 2017) to provide information about virtual resources. This data can be obtained either via the cloud software itself or using external monitoring tools such as Nagios, Ganglia (Massie, Chun & Culler, 2004) or collectd. However, the libvirt plugin for Nagios, nagios-virt offers only the power state of the VM and cannot be used for the monitoring of virtual environments.

Ceilometer (Ceilometer, n.d), the monitoring and metering component of OpenStack, provides data about virtual machines, number of images uploaded by users and their size, amount of block and object storage consumed and the amount of packets sent and received in the network interface. It was introduced only in Havana and there was no stable monitoring support in its previous versions. OpenStack is also compatible with monitoring tools such as Zenoss (Zenoss, n.d) and Nagios. ZenPack, a Zenoss extension (Zenoss, n.d), allows monitoring of flavours, images and servers that are running in the OpenStack clouds. Similarly, Nagios also provides a plugin using which OpenStack services can be defined, configured and monitored. Monitoring driver is the component in OpenNebula (OpenNebula Project, 2017) that is responsible for collecting information about physical and virtual hardware resources. It executes a set of probes in the hosts and the information is transferred to higher-level components using events or interfaces.

Monitoring in VMware vSphere is handled by several tools, which gather and display system information and resource usage. These tools can be accessed by either GUI or command line. Additionally, they support configuration of alarms, setting up alerts and notifications and the necessary actions to be performed when the threshold specific to a particular resource is breached. CloudWatch (Amazon Web Serices, 2017) is a service in Eucalyptus that collects monitoring information from the cloud resources, pre-processes them and converts them into readable metrics. Furthermore, it provides options to configure alarms based on the generated events / data and allows publishing of new metrics in the CloudWatch system. By default, they monitor the following resources: instances, volumes (block storage), and load balancers.

CloudStack (The Apache Software Foundation, 2017) uses Usage Server, which creates a summary of usage records by taking data from the event logs. The interfaces on usage records accept user, project, start and end date as input and return information such as the VM run time, their resource utilization, number of public IP addresses belonging to the user and number of snapshots uploaded. Zenoss also provides an extension, ZenPack, for monitoring the software and hardware resources running under CloudStack. Phantom, the latest release of Nimbus, contains a package tcollector, which provides sensor-monitoring information about the deployed virtual machines. It uses OpenTSDB (Time Series Database) for storing the gathered data that are received from the collectors. Similar to other cloud software, tcollector also collects information about the virtual hardware resources such as processors, disks, networks, processes and the NFS storage.

2.4.3 BASMATI Resource Monitoring Requirements and Specifications

BASMATI platform will provide the federated monitoring information with multi clouds as well as the monitoring data for each cloud and services, so thereby allows the related components in BASMATI platform to make use of the cloud infrastructure and the deployed services. Furthermore, this monitoring functionality should be as cost effective as possible in order to gather information of all running service instances.

BASMATI platform will be interested in monitoring various aspects of each cloud, e.g. current virtual machines running state including CPUs, memories, disks and networks, active SLAs, possible SLA violations, log, and so on. Some of the monitored data will be displayed via a graphical user interface.

For the resource monitoring of all existing cloud services, BASMATI platform just will make use of monitoring services that each cloud services expose to cloud users as well in one form or another. However, BASMATI platform will use resources from multiple cloud service providers that could have their own monitoring system, therefore it should provide a comprehensive and federated monitoring methods that can operate seamlessly across cloud domain borders. In addition, BASMATI

The requirements of the resource monitoring may be as follows;

- Monitoring the hosts, the virtual machines, and the dynamic parameters like the current CPU, memory, I/O and network utilization of a virtual machine from monitoring agents
- Monitoring the static parameters like user- and project-related information and the hardware layout of the virtual machines from cloud and virtualization software
- Integrating multiple clouds, virtualization software, and monitoring tools
- Avoiding bottlenecks due to the communication with respective infrastructure monitoring data supplier
- Considering billing which may be done based on the monitoring data obtained from various cloud layers.
- Considering a general and easy-to-use management method of monitoring data
- Defining a well-structured data model, standardized interfaces, and common databases in order to support the heterogeneity of the federated cloud and providing monitoring data in a meaningful way
- Providing on-demand and on-schedule data management

2.5 Situational Knowledge Extractor

The Situational Knowledge Extractor and Acquisition is the process that takes as input the current state of the application and user behavior and combines it with previously observed pattern behaviors in order to predict the resource needs of the user-application session. The whole process exploits state of the art algorithms and methods providing efficient and efficacy results in real time satisfying the quality of services and service level agreements.

This section presents the process for data acquisition and refinement, clustering, classification, and regression predictions of the resource demands, and finally the requirements for implementation in BAMATI.

2.5.1 Data Acquisition and Refinement

The combination of the various sources of information requires the use of Data Fusion (DF) techniques (El Faouzi & Klein, 2016). DF techniques can combine in a unified fashion all available data derived from various sources providing a vector representation for each user-application session (Girdhar et al., 2016). An alternative representation such as an ontology representation (Liu et al., 2016) will be examined in case that they can represent the obtained knowledge in a more flexible and accurate fashion.

The abovementioned representations may suffer from the curse of dimensionality (Ye & Sugihara, 2016). To mitigate this problem, it will be investigated feature selection techniques (Lou et al., 2015) that will filter out the vector dimensions that decrease the accuracy of the resource predictions. The feature selection criteria will be applied to the annotated datasets that will be provided by the end users in order to gauge which dimensions should be discarded. So, the training dataset can be reformed and the knowledge acquisition can be refined before it is processed by the following stages.

2.5.2 Clustering, Classification, and Regression Predictions of the Resource Demands

The representation of user-application sessions will be used for the resource demands. Three different approaches will be described and researched in order to resolve the emerged challenges of the situational knowledge extractor and acquisition.

Users may have similar behaviors during the use of applications, they may follow the same trajectories or make similar operations like copy, record or move the point of view of visual workstations. An unsupervised clustering (Zare Mehrjerdi & Nadizadeh, 2016) of the users who interact with applications can predict the group of users who will have the same behaviors or users that interact the one with the other. The application that will be used by these users should be serviced by the same server cause that these applications need common data.

Classification of user – application sessions (Taravat et al., 2015) will be a method that will recognize patterns of behaviors that have been observed in previous events. If the sessions have high similarity with the representation of predefined session categories then we can infer the evolution of the resource demands. A set of classes that will be determined based on the annotated datasets. Each class will represent the observed behavior of the user and application. In addition, the classes will provide the observed resource needs based on the previous events.

The Regression (Gu et al., 2015) of the user – application sessions will be a more precise method that can predict the resource needs based on a function that has been gauged by the annotated datasets. The vector representation of the sessions and the infrastructure of the system will be

the input of the regression function. Different use of the Basmati platform may have as a result different configuration parameters of the regression functions. These parameters will be gauged based on the annotated datasets.

2.5.3 Knowledge Extractor Data Management Requirements, Specifications and Further Processes

The difficulty in translating user-understandable application terms from the software application to the resource demand management can be carried out by an artificial neural networks approach (Kousiouris et al., 2011) that takes into consideration the application-level workload parameters, the quality of services and the key performance indicators in order to request the appropriate hardware resources.

The ability to parallelize and process large scale computational tasks on big data is an issue that will be emerged in the Basmati applications. A mechanism for offering virtual clusters as a service (Kousiouris et al., 2013) to address the needs of dynamic and on-demand creation of the data will be provided taking into consideration the time-step evolution of the applications and the user behaviors.

The Study of user's and applications' behavior in the management processes of Clouds can be enhanced by analyzing information at a high-level related to application terms (Translation level) while it predicts the anticipated user behavior (Behavioural level). The identification of patterns in high-level information through a time series analysis is translated to low-level resource attributes with the use of artificial neural networks in a dynamic fashion (Kousiouris et al., 2014).

2.6 Data Management

In order to delve into the topic of data management and governance, we need to define some terms that will ease the reading and understanding of the rest of this section. Firstly, we have to mention that by Data Store we define any means of data storage, be it a Database system, a virtual data storage platform or a physical data storage infrastructure. To move on to a couple more complicated and easily confused terms, we will define the Data Management. According to DAMA International, Data Management is "the development, execution and supervision of plans, policies, programs and practices that control, protect, deliver and enhance the value of data and information assets" ("Data management", 2016). Data governance, on the other hand, is "a control that ensures that the data entry by an operations team member or by an automated process meets precisely standards, such as a Business rule, a data definition and data integrity constraints in the data model" ("Data governance", 2016). As we can see, Data Management is a wider term, covering a wide range of processes and models applied to the retrieval and storage of data whereas data governance is a set of rules, ensuring that the standards set by the users of the data are met.

2.6.1 CoherentPaaS Component Modification

The BASMATI Unified Data Management Framework (BUDaMaF) will be based on the CoherentPaaS project (CoherentPaaS, n.d.). This project has created a framework of wrapping polyglot database system under one common transactional engine, using one common SQL-like query language, even on databases that do not support transactional properties or SQL syntax. Of course, in order for these components to fit into the BASMATI goals and architecture, some modifications may be necessary, especially if we need to add additional functionality. Three main components will be used in the creation of the BASMATI Unified Data Management Framework:

Common Query Language

The language used in CoherentPaaS is officially called "Cloud Multidatastore Query Language" (CloudMdsQL). This language is SQL-based so its syntax and vocabulary is borrowed from the SQL. In a polyglot world though, the need of accessing noSQL data stores necessitates the addition of some Python elements in order for the CloudMdsQL to support embedded native queries and embedded procedural language concepts (Kolev et al., 2015).

Query Engine

The Query Engine developed in the context of CoherentPaaS is an open source software solution that can form the basis for the BUDaMaF transactional manager. It is created with a highly distributed, cloud architecture, taking advantage of the unique environment of a computer cloud.

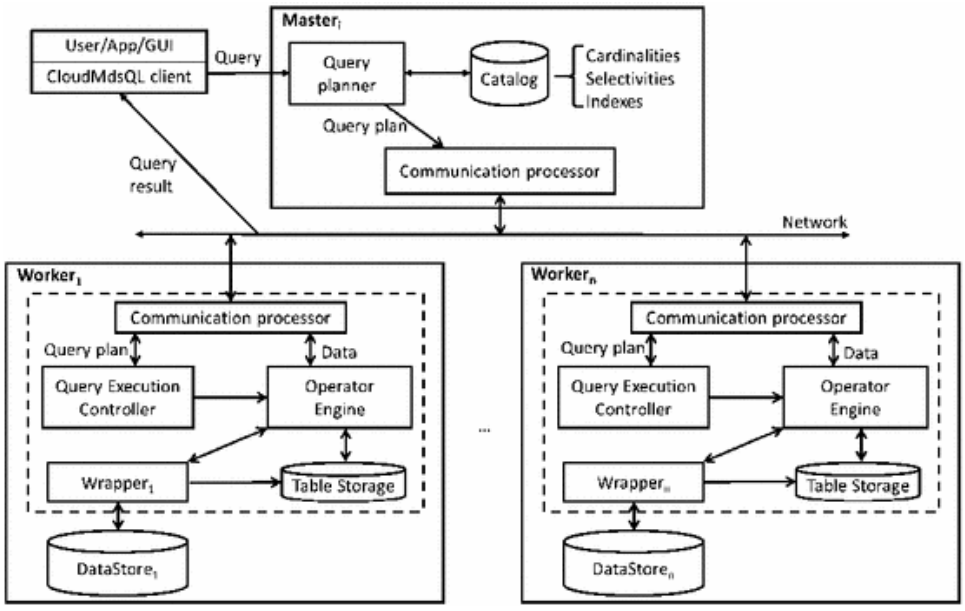

Figure 13: Global Architecture of Query Engine [source: Kolev et al., 2015]

As shown in the architecture, the query engine is developed using a master-slave model, where the master is planning the queries received by the client and then sends them to a worker. This

worker then optimizes the query plan and performs a combination of local calls and data requests from other workers, if needed, in order to perform complex queries in a resource efficient manner.

Wrappers

As presented by Koley et al. (Kolev et al., 2015) the wrappers are developed in the form of plugins to the query engine. They handle the interaction with the local data stores and the translation of CloudMdsQL queries into local queries. Specifically, the wrappers are responsible for four tasks: a) execution of local sub-queries, b) transformation of data into the form expected by the client, c) data delivery to the operator engine and d) instantiation of other named expressions in the local table storage, which works like a cache memory.

2.6.2 Logic and Data Separation

The highly distributed BASMATI architecture necessitates the separation of data from the business logic of the hosted applications, in order to achieve higher scalability, lower resource demand and thus lower costs. This gives rise to a set of implementation challenges, both in the creation of new modules and in the integration of older modules in the BASMATI platform.

When creating new services it is simple to develop them with this separation in mind but when dealing with older and perhaps locked under copyright laws software components it is harder. A possible solution is to create interfaces or even wrappers in order to translate the local data calls of a service into other calls, either local or remote based on the application needs.

2.6.3 File System Support by BUDaMaF

As discussed in an earlier section the BUDaMaF will be based on the CoherentPaaS project. This project has already developed and tested usecase based on MongoDB, which is a noSQL, document based system. It also provides support for other file based systems like Hbase, which is a Hadoop file system data store.

A transition to another file system is a simple matter of wrapping the file system in a more platform friendly form. This can be done in one of two ways, either the file system is converted into a noSQL database and then accessed by the BUDaMaF or a wrapper is created that will translate the commands issued by the BUDaMaF into file editing commands.

2.6.4 Data Located on Mobile Devices

Some of the data used in BASMATI applications will be located in the end users' devices, be it laptops, smartphones or other micro-computers. This creates an issue for the unified framework. It can manipulate the integrated databases seamlessly but it has limited to no access on the end user devices.

Two solutions are proposed for this challenge; the first is keeping duplicates of the relevant data that will be kept synchronized to the actual data into a multi-cloud controlled database. This solution, of course, is increasing the cost exponentially but it is ensuring the data availability.

The second solution is requesting access to the data by the end users, thus creating a connection between the user's device file system and the BUDaMaF. This connection will be using its own wrapper and work in a loosely coupled fashion. Of course, an intermediate solution can be achieved by employing the cloudlets as a mediator layer. The data could be accessed through the cloudlet devices and then managed by the BUDaMaF, which will already have access to the cloudlet data.

2.6.5 Raspberry Pi used for Bluetooth Beacons

Creating a database of trajectories is imperative in order to understand user mobility in a big event, like Das Fest. A solution proposed is creating a network of Bluetooth beacons that will act as checkpoints, logging each user's ID and time of arrival and departure as the users are passing through these beacons. This will create a rough trajectory with timeframes for each user in the area.

In order to minimize the work needed and the cost of implementation, the usage of Raspberry Pies as beacons was proposed. This requires a detailed analysis of each application run in the Raspberries in order to avoid overloading them with resource demands. If using them as beacons is endangering other services running on them, optimization of the services must be performed and if that fails another solution must be explored.

2.6.6 Data Replication and Consistency Mechanism

As a failsafe to the prediction model and self-learning adaptive mechanism, presented in the research challenges, a data replication mechanism is proposed. It will consume more resources, thus increasing the cost, but it is a tested way of maintaining the QoS requirements across different nodes of the multi-cloud. It ensures minimum effort by the applications for accessing and manipulating the data and creates a more solid data structure, also ensuring the data availability at all times. In this case, though, we will encounter a lot of replication and possibly data consistency issues that need to be countered. The OPTIMIS project (OPTIMIS, n.d.) has created a toolkit that deals with data replication issues on large cloud platforms so it could form a basis for tackling this challenge.

2.6.7 Predictive Mechanism for Cost Efficiency

A mechanism for predicting the costs of moving, accessing and editing data for each application must be created. This mechanism will function as a basis for data management decisions. It could dictate, for example, when and where to move the data and how much of the data related to an application to move. For example, an application could use a large chunk of data during its initialization and never use them before. This big chunk of data does not need to follow the application that could be moving between different nodes of a cloudlet. So the data that will be following this specific application are considerably smaller in volume and thus the cost of moving them is lowered significantly.

It is hard to acquire a priori knowledge of each specific application's needs regarding the data so we have to rely on estimations and predictions based on our expert knowledge of how the application works in addition to a detailed analysis of historical data. In addition, the cost of moving the data or having an application access the data remotely also have to be estimated according to the SLAs signed with the users and the infrastructure providers. On one hand, moving the data will require extra resources that will increase the cost of infrastructure usage, on the other hand, accessing the data remotely could lead to QoS violations, increasing the cost towards the end users.

Tackling this challenge, according to these factors, necessitates the creation of two models, one for estimating the data needs for each application and one that estimates the costs for each data regarding the action. These models will work in unison, creating a set of scenarios for data handling for each application with a prediction of the cost that each scenario will impose on the service provider. The scenario will include the volume of data that need to follow the application, the location of the data stores that will be used, the resources needed and the cost of this scenario.

2.6.8 Data Workload Predictive Model

Predicting the data workload of an application is a simple matter of analyzing the historical data and our expert knowledge on how the application is behaving during its life cycle. The main bulk of work is in finding the correct features that will allow us to create a valid and close to reality predictive model of the expected workload. Plenty of work has been done on this field and it seems that two methodologies have emerged in the literature, the Artificial Neural Networks (ANNs) and the Autoregressive integrated moving average (ARIMA) (Calheiros et al., 2015; Fang et al., 2012; Prevost et al., 2011; Roy et al., 2011; Saripalli et al., 2011; Zhang, 2003).

Artificial Neural Networks are a machine learning algorithm, based on multiple nodes (neurons), each neuron performing a fast and simple function. As discussed by (Prevost et al., 2011), these neurons are composed of two parts, one that adds weights to the input data, forming a learning curve, and one that processes the data in a simple manner. The actual algorithm used in resource demand prediction in called Multi-layer perceptron network, in which many layers of neurons, also known as perceptrons, are stacked on top of one another. Each layer's output becomes the next layer's input, always adding more processing power, until a set of predictions are calculated, usually accompanied by their predictive strength, the chance of each one happening.

ARIMA methodology differs largely from ANNs. As (Zhang, 2003) is writing "In an autoregressive integrated moving average model, the future value of a variable is assumed to be a linear function of several past observations and random errors". This means that this methodology is not learning by historical data, it just assumes that the resource demand is following a linear function. So if we know a series of demand values we can calculate the rest by just moving ahead on the time axis of this function.

2.6.9 Cost Predictive Model

In order to predict the cost of any decision regarding the data management plan in regard to an application, we have to take into account multiple variables. In their work, Roy et al. (Roy et al., 2011), they have created an equation that takes into account the three most important of these variables; the cost of SLA violations, the cost of leasing a virtual machine and the cost of reconfiguring a machine.

Based on their equation we can create a similar one for calculating the cost of certain data management decisions. For example, we could calculate the cost of moving a certain percentage of the data required by an application to an infrastructure that is more easily accessible by the application and then the cost of leaving the data where they are and then compare the two costs. If we generalize this into a multi-objective problem, we can arrive at conclusions showing us the most cost efficient policy for data management for each application instance.

2.6.10 Self-training Error Prediction Mechanism

The predictions generated by the predictive mechanism give us a prior knowledge of the data demands and costs of each application, as noted in the previous section. The self-training mechanism, on the other hand, will provide real-time corrections on this prior knowledge. It will be an online learning system, using unsupervised machine learning that will be trained, in near-real time, by data generated during the application runtime. A promising solution, proposed by (Bashar, 2013) is based on Bayesian Networks in order to create an online learning model. A more fitting algorithm though used also by (Dean et al., 2012), is called Self-Organizing Map (SOM). It is recommended due to its low resource demand nature, enabling its usage also for streaming data applications, like in the case of BASMATI. SOM is using a network of neurons, much like an artificial neural network, but with a more specific focus, being optimized for fast learning.

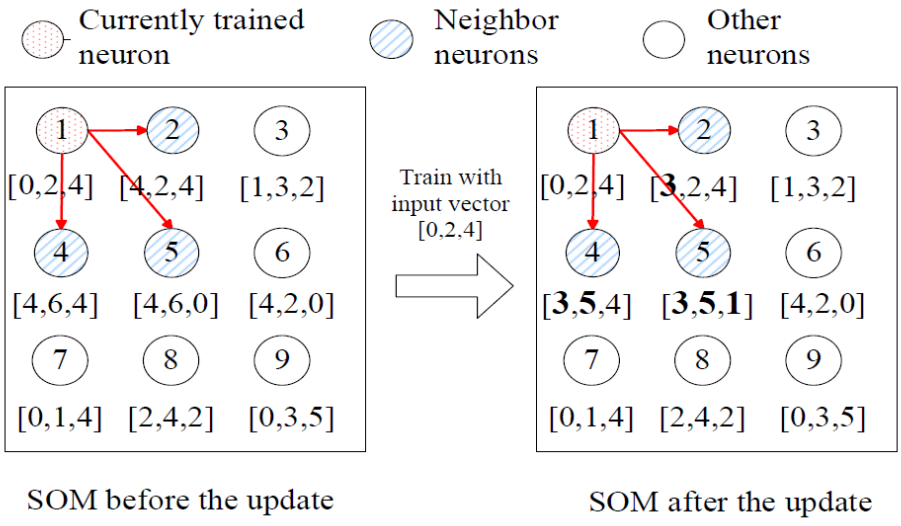

Figure 14: SOM Neuron Training Process [Source: Dean et al., 2012].

This mechanism will have a dual purpose. Firstly, it will provide error detection, in the cases that the prior prediction falls far from the real data gathered during the application runtime, which can be used as a feedback on the predictive mechanism. Secondly, it could change, during runtime, the data policies created during the application deployment. This second function will be available in order to predict errors in the predictive models and try to avoid them. This change in policy of course will not be able to cover actions already taken, like retrieval of data chunks that were deleted or left behind during data relocation, but it could change future actions during the rest of the application's life cycle.

In order to tackle this research challenge, an error location mechanism must be created as well as an online, unsupervised machine learning algorithm and a model that will predict failures of the predictive models. The real challenge is figuring out what an "error" actually is in the context of cost and data usage predictions, finding a list of features that will be directly related to this error locating mechanism and creating a viable model for error prediction in real time and possibly for streaming data. The SOM methodology gives us a viable solution for both locating and predicting the errors by looking at the whole of data provided by monitoring the usage of applications. In order to extract these data though, we still need a monitoring mechanism attached to each application.

2.6.11 Privacy Protection in the Multi-cloud

In the frame of the BASMATI project, the two most common approaches to protect contextual data acquisition from unintended and unauthorized analysis are envisioned: Pseudonymization and anonymization. Both terms are legally defined and refer to the concept of identifiability of a person based on the information given in a dataset. Identifiability of a person is one of the basic characteristics of personal data. "Anonymization" is defined as processing data in such a way that the data subject is no longer identifiable. For the matter of understanding one could add *"to no one"* – so anonymized data is by definition no personal data and therefore not subject to data protection.[2] In contrast, Pseudonymization is the separation of data from direct identifiers so that linkage to an identity is not possible without *additional information* that is held separately. Pseudonymized data stays personal data as is *someone* who is able to map the data to the original subjects that applies to any person who has access to the *additional information*. Flowingly pseudonymized data is still subject to data protection. Ideally, for a person who has sole access to the data – e.g. some service provider or partner processing the data alone - pseudonymized data does not reveal any information that identifies the data subject. Therefore, such techniques significantly reduce the risks associated with data processing, while also maintaining the data's utility. For this reason, the new data protection legislation and especially

[2] See EU directive 95/46/EC (26)

the GDPR recommends and even creates incentives for controllers to pseudonymize the data that they collect.

2.6.11.1 Methods to Generally Anonymize Data

For complex (statistical) data records exist several measures like k-anonymity, n-diversity and t-closeness. The term k-anonymity requires, that each combination of sensitive attributes applies to at least a number of k different data records which implies that there is no combination that is identifying a single data subject (person). The l-diversity measure enhances k-anonymity with regards to so-called homogeneity and background knowledge attacks. The former uses the fact that k-anonymity does not exclude k datasets having identical values of a sensitive attribute. In the latter case, an attacker knows the equivalence class (k) a single subject belongs to. In case this class determines the sensitive attribute. Flowingly l-diversity requires elements of an equivalence class having at l different values for each sensitive attribute. The t-closeness measure enhances l-diversity with regard to cases, in which the l distinct values within an equivalence class are semantically close. E. g. in a collection of health records. In case all values within l describe some serious illness the attacker still receives highly sensitive information. Therefore, t-closeness requires the distribution of attribute values in l having a maximum distance of t to the distribution of the whole table. We refer the reader to see (Danezis, 2005) and (Rai, 2016) for an encompassing presentation and discussion of several approaches. However, all methods to anonymize data introduce a loss of expressiveness by introducing an (increasingly) course classification. So it strongly depends on the concrete scenario if and how they might be applied.

2.6.11.2 Challenges in BASMATI

In Basmati the most important data source is mobility data – i.e. tracking information from users from which at least some additional information is given. Mobility data is required to determine, which computational and storage resources and data are required to serve his needs. These data are potentially sensitive and in addition, introduce some special difficulties with respect to methods applied. Firstly, tracking information is identifying and might hold sensitive information at the same time. Application of k-anonymity and its extensions in the standard approach requires that it is possible to treat both, to some extent, separately. Secondly, background information, which can originate from a variety of other and often openly accessible data sources, is an issue. In complete user tracking records, an identification at any location or point in time or in combination has to be effectively prevented. As a result, the measures to be taken significantly reduce the expressiveness of data and thereby their utility. The same applies to pseudomyzed data, as pseudomyzation is performed in order to keep the *additional information* required for identification secret. If an attacker can infer the person from the data alone, pseudomyzation is impossible. The individual potential to work with techniques like anonymization and pseudonymization depend on the use case and the kind of service in question. However, some general statements are possible:

- In scenarios like adaption of general system resources, it should be considered to use data on (predicted) people frequency and demand.
- Regarding the detection of mobility patterns, individual adaption pseudonymization should be used but combined with an informed consent as a pseudonym is likely to be lifted.
- In the case of an individual service offer to dedicated users of BASMATI neither anonymization nor pseudomyzation can be applied. An informed consent is required.

In any case if possible technical and otherwise organizational precautions have to be applied in order to treat tracking data confidential except for clearly defined and transparent purposes.

Within the BASMATI project, the decision of when and how to use pseudonymization and anonymization will be based on the use case analysis and especially on an analysis of what kind of data will exactly be analyzed and for what purpose. The appropriate mechanisms can be decided only when there are concrete use cases at hand.

2.6.12 BASMATI Data Management Requirements and Specifications

In our case we need both pseudonymization and anonymization, as BASMATI will be a multi-cloud platform having a plethora of users, varying from people to services and mobile devices. So a strict data governance mechanism must be created in order to preserve these data and at the same time a data management engine must be created in order to ensure all the QoS needs that the users demand.

As mentioned, BASMATI will be using a plethora of devices in the form of a multi-cloud architecture. Each type of device gathers or creates data in a format that is not always compatible with each one of the other devices. Of course in order to form a cloud, or a cloudlet, of devices a common data structure and common data storage must be defined, so that the devices that form this cloud can exchange information freely. BASMATI takes that one step further forming a federation of clouds. This federation of clouds will require a common data management solution in order to function in a unified way, as one multi-cloud. This creates a number of challenges, mainly caused by the heterogeneity of the available data and their volume, which a strong data management platform should tackle.

The BASMATI applications will be handling various data from multiple sources and private data will be among them. A protection mechanism must be created and preserved throughout the travel routes of the data. A possible solution lies in the block-chain methodology but needs further analysis before being adapted.

3 Cloud Federation

3.1 Federation Technologies

This section of this document provides the state of the art of cloud federation, or federated cloud.

Several different appreciations will now be given, taken from various public sources on the Internet, to give perspective to the complex subject of cloud federation and to expose the different meanings that these words may take in different companies' minds.

3.1.1 Definition of Cloud Federation

Various research works have defined cloud federation. Haile and Altmann (2015b) described cloud federation as a "strategic alliance between cloud providers, in which cloud providers have reached a cross-site agreement for cooperating regarding the deployment of service components and the use of capacity from each other to cope with demand variations of clients" (Haile & Altmann, 2015b). Another definition is offered by Altmann and Kashief (2014) as " a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Altmann & Kashef (2014).

Apart from the academic works, several IT companies also described cloud federation from their point of view. Two such definitions are given below.

"Cloud Federation refers to the unionization of software, infrastructure and platform services from disparate networks that can be accessed by a client via the internet. The federation of cloud resources is facilitated through network gateways that connect public or external clouds, private or internal clouds (owned by a single entity) and/or community clouds (owned by several cooperating entities); creating a hybrid cloud computing environment. It is important to note that federated cloud computing services still rely on the existence of physical data centers." (APPRENDA, 2016).

"A federated cloud (also called cloud federation) is the deployment and management of multiple external and internal cloud computing services to match business needs. A federation is the union of several smaller parts that perform a common action". (TechTarget, 2011).

3.1.2 Cloud Federation Benefits

"The federation of cloud resources allows clients to optimize enterprise IT service delivery. The federation of cloud resources allows a client to choose the best cloud services provider, in terms of flexibility, cost, and availability of services, to meet a particular business or technological need within their organization. Federation across different cloud resource pools allows applications to run in the most appropriate infrastructure environments. The federation of cloud

resources also allows an enterprise to distribute workloads around the globe, move data between disparate networks and implement innovative security models for user access to cloud resources." (APPRENDA, 2016). Further discussions on the benefits of cloud federation can be found in Kim, Kang & Altmann (2014).

3.1.3 Cloud Federation Implementation

"One weakness that exists in the federation of cloud resources is the difficulty in brokering connectivity between a client and a given external cloud provider, as they each possess their own unique network addressing scheme. To resolve this issue, cloud providers must grant clients the permission to specify an addressing scheme for each server the cloud provider has extended to the Internet. This provides customers with the ability to access cloud services without the need for reconfiguration when using resources from different service providers. Cloud federation can also be implemented behind a firewall, providing clients with a menu of cloud services provided by one or more trusted entities." (APPRENDA, 2016).

3.1.4 Hybrid Cloud vs Federated Clouds

"As cloud computing continues to gain in popularity, people are also looking for ways to make the cloud fit their own personal or organizational needs. As more changes occur, people are trying to decide what kind of cloud they should use, and one of the decisions they must make includes whether to use a hybrid cloud or a federated cloud, or whether to use public clouds. One question that more and more people ask is if there is a difference between federated clouds and hybrid clouds as they seem to be more and more interchangeable." (Smith, 2013).

"A federated cloud is an amalgam of several internal and external clouds that are put together to meet an individual or a business's needs. Just as in the political science definition of federation, a federated cloud is the joining or the union of several smaller parts in order to perform together to accomplish one particular and specific action." (Smith, 2013). "In a nutshell, a federated cloud is a cloud environment that is composed of offerings from multiple vendors — a synonymous term is multi-cloud environments." (Navarro, 2015).

"Over the last few years, many new cloud vendors have entered the market — they offer slightly different services, with different focuses, different hardware, and with data centers in different locations. At the same time, the barriers between these vendors have dropped. Cloud services are more compatible than they ever were previously, and by offering open APIs, they create the opportunity for clients to build bespoke cloud environments that meet their needs better than any single vendor could. Data can be more easily moved between platforms, and integrations layers like computenext make it quite simple to combine offerings and manage federated clouds." (Navarro, 2015)

"Understanding hybrid clouds depends on an understanding of two other cloud modalities: private clouds and public clouds. Public clouds are what is familiarly understand by the term cloud: they are the prototypical cloud environment. Public clouds use virtualization technology

to offer infrastructure and software services to the public. The underlying infrastructure is shared by many different users. Amazon Web Services are a public cloud platform." (Navarro, 2015) "A hybrid cloud is using "the cloud" in a way that an organization or business manages some of the cloud resources in-house, while other resources come from external sources. An organization may use a public cloud service for to archive older data, but they may use in-house cloud storage for customer information that is more up-to-date. Using a hybrid cloud system allows things that are perhaps less important to be stored in the less expensive public sector of cloud computer, while maintaining the more sensitive information and data internally. By keeping the less sensitive material internal, it is more protected from outsiders hacking into the information. (Smith, 2013).

"Ultimately, there is no difference between federated and hybrid clouds. Both terms discuss the joining of internal and outside clouds in order to accomplish one goal. Term federated cloud has been around longer, but as time goes by, the term hybrid cloud has been used more to describe the accomplishing the same task. The terms and hybrid and federated are just different names that describe the same process or existing practice to make it more universally understood by practitioners." (Smith, 2013).

"Private clouds use the same virtualization technology as the public cloud, but the underlying hardware and all the virtual machines and services that run on it are used by only one organization. Typically private clouds are hosted in private data centers or on co-located servers. As you might have guessed, hybrid clouds are a cloud environment that combines both public and private clouds. Often a company will deploy its core computing, and storage infrastructure, on a private cloud and augment that capability with components drawn from the public cloud." (Navarro, 2015).

"It should be clear that although hybrid and federated clouds are not identical, they can be related. It's perfectly possible, and indeed common, for a federated cloud to also be a hybrid cloud — we speak of them as separate categories because it makes it easier to pick out important features. Pundits frequently trumpet the rise of public clouds, or the ascendency of private clouds, as if they were somehow in competition; they aren't. Each vendor and each cloud modality has its strengths and domain of optimal application. The most intelligent cloud strategy is to combine cloud deployment methods, service modalities, and vendor offerings in ways that best meet the operational needs of a specific business." (Navarro, 2015)

3.1.5 Examples of Federated Clouds

The future of the cloud is federated, and when you look at the broad categories of apps moving to the cloud, the truth of this statement begins to become clear. Gaming, social media, Web, eCommerce, publishing, CRM – these applications demand truly global coverage, so that the user experience is always on, local and instant, with ultra-low latency. That's what the cloud has always promised to be.

The problem is that end users can't get that from a single provider, no matter how large. Even market giants like Amazon have limited geographic presence, with infrastructure only where it's profitable for them to invest. As a result, outside the major countries and cities, coverage from today's 'global' cloud providers is actually pretty thin. Iceland, Jordan, Latvia, Turkey, Malaysia? Good luck. Even in the U.S., you might find that the closest access point to your business isn't even in the same state, let alone the same city.

Of course, these locations aren't devoid of infrastructure. There are hosting providers, telcos, ISPs and data center operators pretty much everywhere. If you own infrastructure in one of these locations, you already have a working business model for your local market. And, like most providers, you are likely to have spare capacity almost all of the time.

So, what if there was a way to pool that capacity and make it available as a massive pool of cloud resources to anyone who needs it? That's what federated cloud is all about: capitalizing on this geographically dispersed infrastructure to finally deliver the promise of the cloud.

The federated cloud connects these local infrastructure providers to a global marketplace that enables each participant to buy and sell capacity on demand. As a provider, this gives you instant access to global infrastructure on an unprecedented scale. If your customer suddenly needs a few hundred new servers, you just buy the capacity they need from the marketplace. If a customer needs to accelerate a website or an application in Hong Kong, Tokyo or Latvia, you simply subscribe to those locations and make use of the infrastructure that's already there.

As part of a cloud federation, even a small service provider can offer a truly global service without spending a dime building new infrastructure. For companies with spare capacity in the data center, the federation also provides a simple way to monetize that capacity by submitting it to the marketplace for other providers to buy, creating an additional source of revenue.

There are immediate benefits for end users, too. The federated cloud means that end users can host apps with their federated cloud provider of choice, instead of choosing from a handful of "global" cloud providers on the market today and making do with whatever pricing, app support and SLAs they happen to impose. Cloud users can choose a local host with the exact pricing, expertise and support package that fits their need, while still receiving instant access to as much local or global IT resources as they'd like. They get global scalability without restricted choice, and without having to manage multiple providers and invoices.

The federated cloud model is a force for real democratization in the cloud market. It's how businesses will be able to use local cloud providers to connect with customers, partners and employees anywhere in the world. It's how end users will finally get to realize the promise of the cloud. And, it's how data center operators and other service providers will finally be able to compete with, and beat, today's so-called global cloud providers.

3.1.5.1 EGI

EGI is a federation of computing and storage resource providers united by a mission to support research and development.

The EGI federated e-infrastructure is publicly funded and consists of (as of September 2016):

- 826,500 cores available for High-Throughput Compute
- 6,600 cores available for Cloud Compute
- 285 PB for Online Storage
- 280 PB for Archive Storage

The federation is governed by the participants represented in the EGI Council and coordinated by the EGI Foundation.

3.1.5.2 ONAPP

It can be tempting to think of 'the cloud' as a ubiquitous global phenomenon: always on and always available, everywhere to anyone. And, it's easy to assume that cloud providers like Amazon are the only way you can get access to that kind of global capability. The reality, however, is really quite different. That is why a new approach to the cloud – one based on a federated model – will be increasingly important for cloud providers and users alike.

3.1.5.3 COMPATIBLEONE

The adoption of Cloud computing as a new business model has induced the proliferation of several Cloud service providers. Cloud end users are then faced with choosing the appropriate provider offers in terms of supported technologies, geographic locations, security, access rules, billing, etc. As described above in section 2.4.5 in the context of service deployment, the platform resulting from the CompatibleOne project provides solutions to assist Cloud end users in their provider's choice. The CompatibleOne broker is based on the open standard OCCI, and uses a newly defined object-based description model called CORDS. CORDS serves to model the various Cloud resources that are managed by the main CompatibleOne platform called ACCORDS. This platform is also referred to as the Open Source Cloud Broker since it allows aggregation of all major public and private cloud technology types.

3.1.5.4 EASICLOUDS

EASI-CLOUDS, an ITEA2 project that would provide a comprehensive open-source, innovative and validated cloud-computing infrastructure, the future pillar of this fast-growing market. This infrastructure will feature the three classical categories – infrastructure-as-a-service, platform-as-a-service and software-as-a-service – with superior reliability, elasticity, security and ease-of-use. The infrastructure will be used to set up an application type-specific cloud for private, public or hybrid use and implementing a given level of security, privacy and quality of service. The study of the feasibility of Cloud Federation was an important aspect of this project and resulted in a prototype for the technical deployment and delivery of application resources over

an embryonic cloud federation based on Open Source cloud technology from the French National FUI CompatibleOne project.

3.1.5.5 MICROSOFT

Microsoft uses the term federation simply as a commercial hook to bring attention to their Microsoft Windows Azure Cloud Services.

3.1.5.6 GOOGLE

Similar to Microsoft Google uses the term federation to bring attention to their Google Compute Engine Cloud Services, a homogeneous global cloud only operated by google.

3.1.5.7 AMAZON AWS

The dominant market position of AMAZON AWS as "THE" major public cloud, with their Amazon Web Service Elastic Compute Cloud, means that their provisioning systems are used by the majority of added value service providers, when preparing their offer of service for deployment to customers and for subsequent integration in eventually federated solutions.

The recent announcement relating to the availability of VMWARE based solutions through the AMAZON commercial framework shows that commercial and business developments are currently underway in this area of cloud federation.

3.1.5.8 BEACON

The BEACON project a framework with a cloud federation capability, which enables applications to be setup in cross-cloud scenario (BEACON project, 2017).

3.1.5.9 SUNFISH

SUNFISH plans to develop a secure cloud federation environment with special emphasis on services in the public sector (SUNFISH project, 2017).

3.1.6 Analysis

The remainder of this section of this document will be devoted to the analysis of this information and the different points of view that are exposed with an aim to formulating the BASMATI definition of CLOUD FEDERATION. Two important points need first to be clarified to provide a reference by which the various descriptions above may be compared.

Federated Cloud or Cloud Federation?

Firstly, it is necessary to look carefully at the wording used throughout the different descriptions above where the terms Cloud Federation and Federated Cloud are used synonymously. This feels uncomfortable in that the Object of each expression is clearly very different. In the first case the Object being the Federation and in the second case the object is the Cloud. The first case really signifies a Federation of Clouds whilst the second is more concerned with the idea of a Cloud that is the result of Federation. Whilst the term Cloud is often used to obscure the boundaries of the data center oriented modern computing paradigm, the idea that many different Clouds exist is now widely recognized and as such the term Federated Cloud feels out

dated and will be taken to refer to a Single Cloud that is indeed the result of the Federation of different parties. This is important because the problems that can arise when discussing the subject of Cloud Federation, and of course Federated Cloud, involve the way in which partners can operate not only together but also alone, hence the term Federation, where individual partners, retaining their individual identities and operational scope and abilities, enhance the technical and commercial range of their individual offer by entering into the partnership agreement of the Federation.

Cloud Brokerage or Cloud Federation?

Secondly it is necessary to look carefully at the difference between the terms Cloud Brokerage and Cloud Federation in order to be able to clearly define the relationships between different actors within a complex commercial scenario. Although both terms clearly involve the concept of commercial partners working together within the field of Cloud computing, the first relates to the Client facing situation where a Broker is responsible for the procurement of application resources for their customers and in doing so will make use of secondary cloud partners as appropriate to satisfy the customers' requirements. In order to qualify as "Brokerage", this should be performed in a transparent manner (cf. NIST Cloud Definition) such that the customer is perfectly aware of the identity of the cloud operator, or cloud operators, that will be involved in providing and managing their provisioned application resources. In contrast, the second term, Federation, relates to the Business to Business relationships that exist behind the scenes and has no particular obligation of transparence with respect to the eventual customer since each partner within the federation would be effectively and contractually responsible for the deployment and management of the provisioned application resources made available to their customers.

3.1.7 BASMATI Federation Requirements and Specifications

The BASMATI project intends to make use of computing infrastructure, available in both back-end cloud data centers and front end edge computing environments, to perform dynamic migration and offloading of deployed applications, and their operational components, for use on mobile devices by mobile end users. The emphasis is on end user mobility while maintaining compliance with declared and SLA governed performance criteria. Application end users will launch their applications through the BASMATI platform and, over time as users roam, the applications may be migrated as required to ensure compliance. Migration will be primarily geographical, to ensure the best performance over the so called last mile. Efficient geographical migration is not however possible using one single cloud provider alone, since economic constraints mean that cloud operators simply cannot afford to have data centers in each and every town of every country. This requires that migration must be performed across technological boundaries, from one cloud operator to another, from one cloud technology type to another. To alleviate the problems that both geographical and technical cloud migration raise, BASMATI intends to base its cloud provisioning on the concept of the CLOUD FEDERATION. This federation will comprise partner operators, each of which will specialize in particular

technical or geographical cloud domains. The federation members will comply with centralized federation protocol as defined by the BASMATI project and will allow individual federation members to commercialize their services to their customers alone and in collaboration with the other federation members. This concept has been touched upon by both ComputeNext and ONAPP in the overview of the current state of the commercial art of cloud federation. Both companies provide proprietary commercial solutions for the aggregation of data centers and cloud providers. The approach of BASMATI will take this one step further providing the methods and tools, for the construction of a federation of cloud service providers, and cloud operators, that are able to seamlessly aggregate not only private data centers and public clouds but also the new generation of edge computing environments and their associated providers. Although, not surprisingly, the major Public cloud operators, AMAZON, MICROSOFT and GOOGLE, firm in belief of their own individual strengths make no claim or reference to this degree cloud federation, they will probably become the corner stones, and work horses of such future cloud federations.

The BASMATI cloud federation will provide solutions in the form of both the tools and methods to the following problem domains.

3.1.7.1 Accounts, Costs, Billing and Identity

- Each member of the cloud federation will cater to the needs of their own customers. As such each member will have cloud provisioning subscriptions with the various cloud providers with which they work for the placement of their resources and will be responsible for the costs incurred for the operation of those resources.
- Each member of the federation will be responsible for the billing of their own customers based on the service level agreements and associated pricing models that they practice with their customers.
- Members of the federation will be able to request deployment of resources by other members of the federation for use when building application solutions for their own customers.
- The federation member requesting deployment by another federation member will be responsible for payment of the costs incurred as described in the service level agreement between the different federation members.
- The notion of a standard federation level identity is very important in order to ensure transparency and interoperability between the members of the federation at this very important financial level.

3.1.7.2 Application Modelling

Generic application modelling constructions should be shared between the different federation members, or at very least provision must be made for the appropriate real-time conversion of description domains between the different members.

3.1.7.3 Agreement Terms

Service level agreements will have been negotiated and accepted by the various stake holders that are participating in the federation scenario.

3.1.7.4 Resource Placement

The resource placement decision is the moment when federation extensions may be selected in order to satisfy a customer's application resource requirements outside of ones' own domain of competence.

3.1.7.5 Resource Deployment

Resource deployment across the federation will be handled by the appropriate member of the federation that has been selected during the placement procedure.

3.1.7.6 Resource Monitoring

Resource monitoring and subsequent decision taking will be under the responsibility of the service level agreement manager of the corresponding federation member through which the customer has initially been in contact. This is important in order to ensure that monitoring flows through the different federation members do not overtly impede their own local operations.

3.2 Economic Models of Cloud Federation

Based on Buyya et al. (Buyya, Ranjan, & Calheiros, 2010) Cloud computing is "network of virtual services" which allow end users to access application services from anywhere in the world on demand. It delivers infrastructure, platform, and software at competitive costs depending on users QoS (Quality of Service) requirements without large capital outlays in hardware and software infrastructures. Nevertheless even biggest Cloud service providers (CSP) cannot establish data centers at all possible geographical locations. For example, Google has three point of location data centers in US (mostly in the East coast), in the north part of Europe and in Asia (Singapore and Taiwan) (Google Data Centers, 2016). So in some cases CSP may not provide service that meets QoS because of user's multiple geographical locations (Risch & Altmann, 2009a).

Worse situation have small data centers: no ability to dynamically expand capacity, to offer market competitive price and to provide QoS aware services. All they have is limited local market that to not give them opportunity to maximize profit and grow. Only way to solve this issue, in what all scholars unanimously agree (Buyya, Ranjan, & Calheiros, 2010), (Altmann & Kashef, 2014), (Kim, Kang & Altmann, 2014), (Haile & Altmann, 2015b), (Kurze et al., 2011), is to share their resources and clients and build so called Federated Cloud. Cloud federation contains services from different providers aggregated in a single environment enabled by interoperability features such as resource migration, resource redundancy and the combination of complementary resources. So users get profit from lower costs and better performance, when CSP can offer more optimized services (Kurze et al., 2011). In (Kim, Kang & Altmann, 2014) the authors compare big CSP with a federation of small CSP based on economies of scale and

network externalities in an oligopolistic environment. The results of the paper provide a guide for designing economic models for Cloud federation to analyze their sustainability.

On the other hand, sharing resources brings out not only the technical aspect of integration process but also economic (Haile & Altmann, 2015a; Haile & Altmann 2016). It is not possible to write complete contracts, i.e., contracts containing all, even future aspects of business relations (Kurze et al., 2011).

To begin the study of economic-based methods of Cloud federation lets discuss the economy of a single Cloud. Early approaches with different types of pricing strategies were presented in a various papers (Samimi, Teimouri, & Mukhtar, 2014; Weinhardt et al., 2009; Weiss, 2007), (Altmann, Courcoubetis, & Risch, 2010) which distribute cloud resources to users. These methods can be subdivided into fixed methods and dynamic methods. The most popular method, which is implemented by major CSPs (Weinhardt et al., 2009), (Armbrust et al., 2010; Al-Roomi, Al-Ebrahim, Buqrais, & Ahmad, 2013), is the "pay-per-use" method. Each resource is assigned a fixed price. In the case of the subscription model, any number of cloud resources can be used for a fixed price for a certain period of time. In (Hamsanandhini & Mohana, 2015), the authors consider the "pay-per-use" method to propose a set of policies that allocate VMs according to user's QoS purchased. However, the authors only discuss cloud resources as a virtual pool of the physical infrastructure and do not go deep into resource specification. Al-Roomi et al. used dynamic pricing to allow CSPs or users to change the price depending on pertinent factors (Al-Roomi et al., 2013). Auctions are also dynamic methods (Wang, Tianfield & Mair, 2014). For example, Amazon EC2's spot instance is a sealed-bid uniform price auction. These economic-based methods are only applied to the management of external resources usage.

Negotiation between CSPs should consider a wide-ranging number of parameters, such as the provider's incoming workload, the cost of outsourcing additional resources, the revenue for renting unused resources, or the cost of maintaining the provider's resources (Altmann, Hovestadt & Kao, 2011; Goiri, Guitart, & Torres, 2012). To solve this issue Goiri et al. presented an approach where CSPs make decision based on maximization of expected profit in 5 possible situations:

- Run VMs using local resources.
- Run VMs on federated providers (outsource).
- Offer idle resources to other federated providers (insource).
- Turn on off nodes.
- Turn off idle nodes.

Authors obtained equations that establish the relationship between the utilization and the capacity for provider's profitability. Where CSP is profitable when revenue is higher than the capital costs (CAPEX), and the operational costs (OPEX) (Altmann & Kashef, 2014).

To analyze cost model in more detail (Altmann & Kashef, 2014; Kashef & Altmann, 2011) presented an algorithm for making service placement decisions in federated hybrid clouds. Authors identified 21 different cost factors structured and categorized into six main groups: electricity, hardware, software, labor, business premises, and service. To find the optimal service placement presented algorithm calculates the total cost for each possible service placement option on a private cloud and a number of federated public clouds. And as the main outcomes authors underlined two issues that need to be considered closely in federated clouds: (1) the deployment cost has a strong impact on the optimal service placement, (2) the data transfer cost is a significant cost factor in federated clouds (Altmann & Kashef, 2014).

As another idea for implementation of economic models in distribution federated computing resources (Altmann, Courcoubetis & Risch, 2010; Altmann et al., 2008; Majhi & Bera, 2014) present to integrate an auction (bidding) in the VM migration process. Authors developed the algorithm of the interaction of CSPs in VM marketplace based on English and Dutch auctions with describing components as actors, relations, and business model.

Federation models can also be based on autonomic computing engine (Breskovic et al., 2011). The paper on CometCloud by Petri et al. (2014) differentiates task in the Cloud as the number of workers allocated to local and external/remote requests. When one CSP cannot to process tasks from its local users within their deadlines it negotiate for the outsourcing of tasks to other CSP. The negotiation considers the cost for local jobs and cost for remote jobs. As well as polices must be fulfilled as the local task is always accepted first or remote task is accepted if the price is higher than cost.

3.3 BASMATI Federation Business Requirements

BASMATI aims to provide a set of tools for the development, deployment and configuration of federated cloud environments from multiple platforms and providers. As was mentioned above there is no one standard economic method of distribution of virtual resources. So BASMATI will have to face the issue of interpretation of socio-economic criteria from the complete ecosystem. As a dynamic system of supporting the decisions through runtime adaptable deployment patterns of the brokerage platform, BASMATI cannot assign a fixed pricing method. Continuously change of prices that member of the federation accepting for providing the resources leaves no opportunity to set simple pricing plan. Moreover, large events, with thousands of visitors and their mobile devices, such as festivals, sports events, and the like bring the risk of demand more resource, which in certain cloud provider hard to provide. Since, it is difficult to assume that there will be a big number of members of the federation from the same geographical area, any type of bid-based method with different rules of the auction can be immediately dismissed. Because to maximize the total social welfare by using auction methods a marketplace with a big amount of players interesting in purchasing a service or good is required.

Negotiation between members of BASMATI should consider a wide-ranging number of parameters that accepted between the various stakeholders that are participating in the

federation. And tree simple rules have to be underwritten: (i) local tasks have to be deployed first, (ii) deployment is implemented only if cost<price, (iii) replacement is implemented if only current profit<new profit.

As one of the ambitions of BASMATI is to define innovative approaches for describing the application usage and modes a new concept of the economic model should be applied in the project. Besides most of the method presented in the literature review consider a limited number of socio-economic parameters. BASMATI, in its turn, aims to cover a broad set of the features such as the heterogeneity of resources, ultra-scalable resource provisioning, computing offloading support, data integrity issues, context- and situation detection, quality of service and protection assurance, user- and application modeling. It requires to establishing adequate Provider SLAs and Mobile SLAs, which assure that both applications and infrastructure meet the promised performance benchmarks. Detailed list of parameters described in those SLAs will support the exploitation of heterogeneous and dynamic economic model already available solutions.

3.4 Federation Topologies

Many approaches for intercloud aimed to the creation of unified access points that act as a gateway toward the whole cloud infrastructure resources. According to (Altmann & Kashef, 2014, Kashef & Altmann, 2012; Grozev and Buyya, 2014), intercloud approaches can be categorized as centralized with a central, often logical entity that works access point, or flat, in which any cloud datacenter can work as an access point.

3.4.1 Centralized

InterCloud (Buyya, Ranjan & Calheiros, 2010) is a federated cloud computing environment that enables a common marketplace in which applications are negotiated among brokers and cloud providers. InterCloud performs application scheduling, resource allocation and migration of workloads. Intercloud is built on three concepts: Cloud coordinators, Cloud Brokers and Cloud Exchange. A Cloud Coordinator (CC) exports the services provided by a cloud to the federation by implementing basic functionalities for resource management such as scheduling, allocation, workload and performance models. CCs periodically update the Cloud Exchange (CEx) with their availability, pricing, and SLA policies, which in turn aggregate information supplied by CCs in order to support the Cloud Brokers activity. The Cloud Broker identifies suitable cloud service providers published on the CEx, negotiating with CCs for an allocation of resources that meets QoS needs of users.

The CONTRAIL (Carlini et al., 2011) approach is based on a conceptually centralized mediator between Cloud users and providers, which offers resources belonging to different Cloud providers to users in a uniform fashion. Contrail focuses both on the vertical and horizontal integration of multiple cloud providers, organizing the enforcement of QoS by the definition of federation-level SLAs, which also drive the resource selection process and can be mapped on single cloud providers SLAs. The vertical integration is realized by means of the unified platform

to access the different flavours of resources provided. The horizontal integration is performed by the interaction of the different Cloud providers.

Federated Cloud Management (Marosi et al., 2011) is an inter-cloud architecture that provides the integration of multiple IaaS. It is based on a logically centralized component called Generic Meta Brokering Service (GMBS) that receives requests from the cloud users and redirect them to the IaaS chosen for the execution.

OPTIMIS (Ferrer et al., 2012) realizes a services toolkit aimed to orchestrate the lifecycle of the applications, which in turn allows the self-management of the cloud federation. It provides users with a platform to search for Cloud services given a set of requirements regarding the allocation of data and computation such as elasticity, energy consumption, risk, cost, and trust. Regarding provisioning models of resources, OPTIMIS provides Cloud bursting, multi-Cloud provisioning, and federation of Clouds.

3.4.2 Distributed

In the distributed models (Altmann et al., 2007; Rochwerger et al., 2010) model, each resource provider is an autonomous entity with its own business goals. A provider can choose the providers with which to federate. There is a clear separation between the functional roles of service providers and resource providers. Service providers are the entities that match the user needs by finding resources that their application needs. However, service providers do not own the resources, but rather they lease such resources from resource providers. RESERVOIR defines a decentralized and distributed architecture for cloud federation in which providers communicate directly with each other to negotiate the utilization of resources. RESERVOIR is based on Claudia (Rodero-Merino et al., 2010) an abstract layer for service management in cloud federations, now integrated with OpenNebula (Moreno-Vozmediano et al., 2012).

Dynamic Cloud Collaboration (DCC) (Celesti et al., 2010) is an approach for setting up highly dynamic intercloud federations. The cloud provider (CP) that wants to setup a federation assumes the role of the primary cloud provider (pCP), whereas the federated cloud providers are called collaborating CPs. To federate new collaborating CPs, adding their resource/services to a DCC platform, an approval of other providers based on their own policies is needed. Users request services published on the service catalogue of the pCP. Then the pCP finds suitable partners based on the business objectives, and stipulate a contract with specific SLAs requirements for each partner involved. If after a distributed negotiation an agreement among all partners is reached a new dynamic cloud became operational.

3.5 Interoperability within Federations

Members of a federation come with heterogeneous data and service description schemes. Interoperability is a fundamental provision in order to manage communication and collaboration within federations. This section provides the concept and issues of interoperability

between multiple clouds, requirements of interoperability in achieving the objectives of BASMATI.

3.5.1 Interoperability and Portability

Interoperability is the ability of a component to work simultaneously with one or more components of other platforms, regardless of the differences between the platforms (Rezaei, 2014; Brennan, Walshe & O'Sullivan, 2014). That means, interoperability allows components, which are simultaneously active on more than one platform, to interact (communicate) to serve a common purpose.

In the case of cloud services, interoperability means that all components (i.e., hypervisor, orchestration, automation systems, and metering tools) can come from different cloud service providers. An interoperable service platform allows for the creation of a system of compatible suppliers. This allows users to compose the best cloud services and to meet their specific needs on a pay-as-you-go basis (Rezaei, 2014; Silva, Rose, & Calinescu). It helps users to compose solutions that are highly optimized towards their needs.

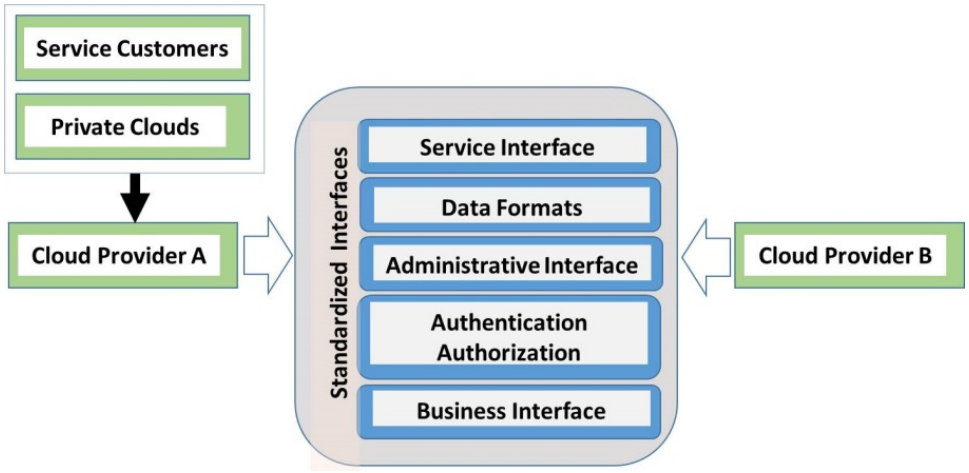

Figure 15: Concept of interoperability [Source: Haile & Altmann, 2017]

Another important and related provision is portability. A cloud service platform, which supports portability, allows data, applications, and/or virtual machine images to be moved easily from one provider to another. Customers should be able to select service providers depending on their performance needs, geographic location, and budgets. Any lack of portability of services hampers flexibility of use, as service providers have no easy way out from a cloud platform if they are dissatisfied with the services offered by a provider and seek a better alternative offered by another provider (Haile & Altmann, 2017). End users of those services (e.g., enterprise users) also benefit from the ability to move their data to a comparable service with less adaptation cost.

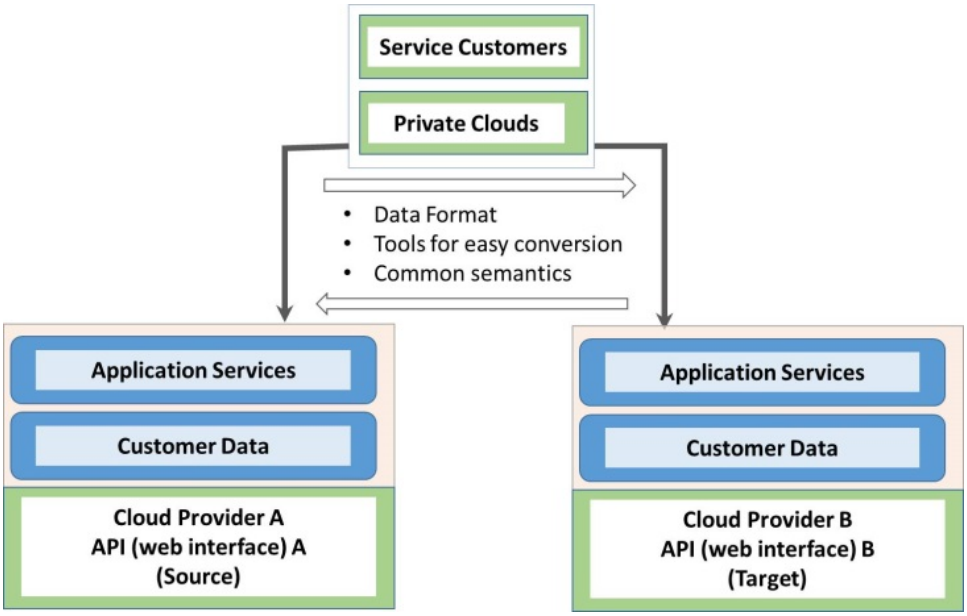

Figure 16: Concept of portability [Source: Haile & Altmann, 2017]

3.5.2 Cross-Cloud Interoperability Challenges

There are many examples that show the lack of interoperability of cloud services. For example, Microsoft provides .NET application containers and Azure database services (Microsoft, 2017), which cannot be integrated with Google App Engine (Google, 2017) and vice versa. Another example is the data formats used by different social media websites (e.g., Facebook), making it difficult to move profile data from one platform to the other. The different technology types with potential issues of interoperability can be classified into programming frameworks, application programming interfaces, and data formats (Gebregiorgis & Altmann, 2015). Consequently, those technology types represent the means for achieving interoperability and portability:

- Programming Frameworks: As service developers are used to their software development tools, programming languages, and runtime environments, they avoid a change of their programming frameworks. Thus, cloud service platforms need to support the programming frameworks that are in use today. Any change of a programming framework would come with the cost of learning a new technology for service developers (Rezaei, 2014; Dustin, Bartlett & Bruklis, 2010)
- Application Programming Interface: Many cloud service platform providers have their own application programming interface (API), preventing orchestration of applications across multiple service providers. The incompatibilities of the APIs for uploading, downloading, inspecting, and configuring are a significant issue for interoperability. For example, the Amazon EC2 API (Amazon, 2017) is different from the Datapipe API (Datapipe, 2017) even though both offer similar IaaS. In addition to this, cloud providers have their own proprietary services. An example of valuable but proprietary services is Amazon Elastic Load Balancing (Amazon, 2017). This service only exists on a specific

platform. Standardization of APIs would allow applications to be moved to different providers without any additional integration or switching cost. Another way would be the provisioning of programming toolsets which enable services to be deployable on multiple clouds (Machado, Hausheer & Stiller; Ortiz, 2011). Although this kind of middleware could break the dependencies on specific APIs, it is only useful for new application developments (Bozman, 2010). Similarly, the format of virtual machines (VM) is one of the major challenges for creating seamless portability among service platforms. As service platform customers have already different virtualization environments (e.g., VMware, Hyper-V, KVM, and Xen) in their in-house data centers, cloud service providers should make sure that customers can use a familiar one (Bozman, 2010).

- Data Format: Many application services provide their own semantic and format for storing data (Rezaei, 2014), making it costly to access the data with a different application. If no standard for the data format exists, conversion tools are needed to translate the data format of one application service into the data format of another application service (Ranjan et al., 2015).

Though interoperability and portability will provide a reduction in cost for switching, learning new technology, and integration, it is to be noticed that incompatibility is a logical consequence of technological development and innovation (Haile & Altmann, 2017). As innovation and technological development results in an increase in revenue, the challenge is to find a compromise between both aspects: cost reduction and fostering new products.

3.5.3 BASMATI Interoperability Requirements and Specifications

The Basmati project intends to provide a cloud abstraction layer that will allow the construction of cloud federations comprising a collection of individual and federated cloud operators. In order to be able to ensure transparent inter-operability of the deployed elements required for automated application provisioning across multiple and heterogeneous cloud providers and technologies, the choice of the base operating system is of utmost importance.

Not only must the same version of the system be available on each provisioning system but also it should be extremely stable over a reasonable period in terms of both its nomenclature and its technical capabilities.

In addition, in order to facilitate the deployment of application resources, the cloud provisioning system must allow the “just-in-time” installation of new generation software configuration agents on the fly without requiring a costly collection of base or golden images to be prepared in advance in each region of each cloud provider for each cloud operators’ subscription accounts.

Hands On experience gained by Amenesik over the past 6 years working with the first CompatibleOne Accords Platform and then the industrial version of the Amenesik Cloud Engine, during the period from 2010 to the time or writing, demonstrates that only the Ubuntu LINUX

operating system is able to satisfy all of the above criteria, and again not only for all major public cloud platforms (Amazon AWS, Microsoft Windows Azure, Google Compute Engine, IBM Soft Layer and Cloud Sigma) but also the majority of the OpenStack and Eucalyptus based public and private cloud operations of RackSpace, OVH, HP, CloudWatt and Numergy.

The LTS versions (Long Term Support) of Ubuntu 10.04, 12.04, 14.04 and more recently 16.04, made available and supported by Canonical for all these environments, have provided not only a stability of nomenclature and technical functionality across the board, but have also ensured upward compatible binary portability for the compiled and linked system components comprising the software packages for the important cloud management functionalities of the various post-configuration, scalability and monitoring agents.

The transition from LTS version to LTS versions is not, however, painless and requires that all software packages be qualified for operation on the more recent versions. This is due to the changes in security and configuration procedures required for many of the major software packages such APACHE, PHP, MYSQL, POSTGRESQL and SMTP, which are imposed by the increasing hostility of the global public internet operational environment.

It is therefore highly recommended that the Ubuntu version 16.04 LTS be adopted as the base operating system for all components developed for Basmati and for the duration of the project.

4 Service Level Agreements

Service Level Agreements (SLAs) play a key role in cloud computing by being the mechanism that users have to enforce guarantees around performance, transparency, conformance and data protection. As cloud adoption increases, cloud users will be seeking more tightly defined SLAs as a means to build up dependable and trustworthy relationship terms with cloud providers (Juan Ferrer & Montanera, 2015). Common concerns with regards to cloud adoption; compliance, security, privacy and integrity, rely on the inability for users to measure, monitor and control activities and operations in Cloud's third party infrastructures (Haile & Altmann, 2015b). In the following sections we analyze state of the art for SLA management in the three different environments applicable for BASMATI project.

4.1 Federation Level Agreements

In a federated cloud scenario, a cloud provider sub-contracts capacity from other providers as well as offer spare capacity to a federation of cloud providers (Altmann & Kashef, 2014). Parts of a service are placed on remote providers for improved elasticity and fault tolerance, but the initial cloud provider is solely responsible for guaranteeing the agreed upon SLA. The federated cloud scenario is related to community cloud set-ups, or from a commercial perspective, for cloud providers that own multiple cloud installations in diverse regions, in order to balance workload among them. A number of research activities have analyzed SLA Federation in this context.

Besides the general problem described above the other big problem not solved today is the language barrier between providers. Each provider uses its own terminology to describe the terms of its SLAs which makes it difficult to impossible for the customer to compare the offerings of different providers and select the most appropriate for its actual service requirements (Breskovic, Altmann & Brandic, 2013). While it is hard for humans to compare and select providers it is almost impossible to automate this, e.g., when trying to organize a federation including multiple providers based on their offerings. Moreover, the current practice requires the customer to manually create individual SLAs with the different providers.

OPTIMIS Project (Ziegler & Jiang, n.d.; Ferrer et al., 2012) delivered an open source toolkit for automated, SLA-driven federation of resources from different Cloud providers. It considered four different cloud deployment scenarios: private cloud, cloud bursting, federated and multi-cloud. OPTIMIS used the concept of Service Manifest in order to express user requirements later included as part of the SLA. These focused on service or provider risk, trust, ecological or cost levels (TREC), as well as legal requirements (when dealing with personal data).

SeaClouds project (SeaClouds, n.d.) investigated the role of SLAs in multi-cloud environments composed of both IaaS and Cloud providers. It defined two levels of SLAs: Customer - Application Provider SLA and Application Provider - Cloud Provider SLA. The Customer - Application Provider SLA represented the SLA between the Application customer and its providers whereas the Application Provider - Cloud Provider SLA detailed the SLA among the Application provider and the IaaS or PaaS provider.

The Contrail EU project (Contrail, n.d.) delivered multilevel SLA management: the final user negotiates a SLA with the Federation, which in turn negotiated SLAs with multiple Cloud providers on behalf of the user, and then, selecting the best offer based on user's criteria (e.g. minimize overall costs); Contrail SLAs are expressed using the SLA@SOI Project defined formalisms (SLA@SOI, n.d.), and extend it in order to support also QoP terms (e.g. storage location).

4.2 Provider Level Agreements

At all levels of Cloud Stack (IaaS, PaaS, SaaS), the requirement of establishing adequate Provider SLAs is to assure that both applications and infrastructure meet the promised performance benchmarks.

Service Level Agreements (SLA) widely applied in research implementations rely on previous works performed by the research community in Grid computing (Risch, Brandic & Altmann, 2009; Quan & Altmann, 2009; Breskovic, Altmann & Brankic, 2013). There are two major standardization efforts available addressing dynamic electronic management of Service Level Agreements (SLA) in order to define Provider SLAs.

4.2.1 WSLA - Web Service Level Agreements

The WSLA (Web Service Level Agreements) specification was published by IBM in January 2003 (Ludwig et al., 2003; Risch & Altmann, 2009b). WSLA is currently barely used due to issues with flexibility of the schema which always requires adaptation to be used in a specific domain and, moreover, due to lack of community support.

4.2.2 WS-Agreement

WS-Agreement is a full recommendation of the Open Grid Forum (OGF) (OGF, n.d.). It defines a language and a protocol to create SLAs. The defined protocol allows both one-step and multi-step negotiation between a provider and a consumer of a cloud service. Usually, it is based on templates of the QoS offerings defined by the provider. While the one-step negotiation is sufficient for a large number of cloud contracts, there are cases that require multi-round negotiation. For these, the process includes a series of offers and counter-offers that allow achieving a SLA agreement that satisfies both parties. This protocol is specified in WS-Agreement Negotiation (OGF, n.d.). In addition, WS-Agreement permits to plug-in specific term languages to define main concepts and particularity of services and its properties for different domains.

An overview of different standards and approaches applied for SLA Management by European Research projects is detailed in (Kyriazis, 2013). This report defines a set of recommendations for further research in the area highlighting the potential and need for standards, encouraging SLA-relevant standards (i.e. OCCI) to add enhancements to enable SLA support as well as to consider overall SLA lifecycle: SLA specification, monitoring tools and the management frameworks.

4.3 Mobile SLA Manager

There are no works on mobile SLA manager available. Therefore this section focuses on the methods of QoS management in mobile cloud.

QoS is recognized as one of the challenging issues for future development of Mobile Cloud together with security, availability seamless mobility and billing (Qureshi et al., 2011). However, already a number of different works have explored different approaches to the issue from diverse perspectives.

Klymash et al. emphasize the QoS challenges in Mobile Cloud due to dynamicity in the available resources (Klymash et al., 2014), among others, at level of bandwidth, delay, packet loss ratio, battery life, and storage capacity. The proposed solution in this work addresses the specific network QoS management challenges of multimedia services and applications in IP Multimedia Subsystems. In order to do so, it focuses on three specific parameters (packet loss, delay and jitter) in order to compare QoS provided by different algorithms for traffic control. Although, the proposed work analyses these on the context of mobile cloud, it does not take into account

specific characteristics for application mobile to Cloud off-loading (as it is the objective in BASMATI).

Differently, from previous work, (Ye et al., 2011) proposes a QoS and power management framework for mobile to service cloud task off-loading. The overall ambition of this work it is to address battery scarcity in mobile devices in order to select the execution platform (mobile device or service Cloud) that allows satisfying user's defined QoS constrains while achieving maximum energy savings. In order to do so, it considers the trade-off among mobile local execution of a task and its associated energy consumption - versus the remote service cloud execution of the same task, therefore, incurring network transmission energy consumptions. The proposed framework considers two operation modes with regards to mobile device to cloud task migration and associated data transmission: full and partial migration. Experimental studies have focused on two QoS attributes: access latency and energy consumption. While BASMATI approach does not specifically focus on the mobile device energy optimization, the approach taken in this paper the trade-off approach among local and remote execution could be of applicability for BASMATI.

Zhang and Yan (2011) propose a Mobile Cloud QoS management framework taking an adaptive approach to QoS management (Zhang and Yan 2011). The framework is designed as follows. Each of the mobile devices has a QoS agent installed that monitors CPU consumption, connection speed, remaining battery and packet loss rate, among others. These QoS parameters are then centrally evaluated at Cloud environment so to decide among different execution models, which dynamically adapt Cloud resources assigned to the service execution.

Yin et al. (2015) present a 2-tier mobile cloud computing framework specifically tackling the needs of Big Data (Yin et al., 2015). It enables pay-per-use and on-demand approach of Cloud instances so to enable multi-tenancy of Cloud virtual infrastructure.

ThinkAir (Kosta et al., 2012) present (see 2.2. for detail description) a framework that enables method level computation off-loading to Cloud environments. The main novelties provided by ThinkAir rely on a more sophisticated use of Cloud computing environment aiming to exploit Cloud potential with regards to elasticity and scalability for Mobile Cloud benefit. ThinkAir provides on-demand cloud resource allocation in order to cope with different requirements of mobile applications to off-load at the level of CPU and memory resources. Besides, it enables parallelization by dynamically managing virtual infrastructure in the Cloud environment, therefore reducing both cloud server's side and overall application's execution time and energy consumption.

ThinkAir QoS management capabilities are being extended in the context of RAPID project. RAPID (López et al., 2016) proposes a new heterogeneous architecture that enables task offloading on the client side and handles offloaded tasks in the cloud side with remote CPUs and GPUs, thought ThinkAir framework. This is enriched with monitoring of the activity of physical

resources in Cloud and support of QoS aspects per each user. These way, ThinkAir implementation, and overall RAPID results will not only realize QoS-based offloading but also create new and more advanced services in mobile applications thanks to remote CPU and GPU accelerators.

4.4 BASMATI SLA Management Requirements and Specifications

In order to tackle the challenge of managing the federation of several CSPs a Federated SLA Management component is required within the BASMATI's architecture. As one federated cloud-based application in BASMATI can have one or more dependencies on underlying cloud service providers, the defined SLAs should include the agreed terms between the CSPs and the application controller (in this case the different SLAs should all be linked to the same application controller). To support the modelling of the SLA hierarchy, in which a higher-level SLA depends on a lower-level one, the implementation at the modeling level could take different approaches. On one hand the lightweight approach, all SLAs are independently modelled and their interactions are just modelled through a particular property in the parent SLA. On the other hand, in a more integrated approach, all SLAs might be modelled in one composite/master SLA.

In any case, to support the federation we might need to consider several aspects in advance, in order to enable the assessment of the contracts and the convenience of the remedial actions at runtime:

- Trustworthy monitoring sources capable of collecting the information of an application spread across a federation of providers.
- The federated monitoring system may need to aggregate the information gathered at different levels in order to evaluate the application SLOs.
- We may need to express complex conditions for the applicability of the SLA.
- Remedial actions should be known beforehand and
- Penalties could be linked to policies associated with several domains: financial, performance or sustainability among others.
- We may need to specify the association between the low-level SLOs violated and the consequences in the high-level agreement.

5 Brokerage

Cloud Service Brokerage (CSB) represents a new type of service and emerging business model in the space of cloud computing, aimed at helping enterprises to address precisely those challenges, and to mitigate the risks that ensue from the complexity inherent in large-scale enterprise cloud environments. In an analogy to the way other kinds of intermediaries operate within different areas of traditional commerce, a Cloud Service Brokerage is an entity that works on behalf of a consumer of cloud services to intermediate and to add value to the services being consumed.

The US National Institute of Standards and Technology (NIST) defines cloud computing in its Reference Architecture for Cloud Computing as a model with Cloud Broker as one of five main roles as: An entity that manages the use, performance, and delivery of cloud services; and negotiates relationships between Cloud Providers and Cloud Consumers (NIST, 2011).

Gartner defines "brokerage" as a model of business; the term is used to refer to "the purpose of a business that operates as an intermediary". More specifically, Gartner uses the term to denote "any type of intermediation that adds value to the consumer's use of a service" (Plummer, 2011). According to the same analysts, a business cannot be considered a Cloud Service Brokerage if it does not have a "direct contractual relationship with the consumer(s) of a cloud service" (Plummer, Lheureux & Karamouzis, 2010).

According to Gartner (Plummer et al., 2011), Cloud Service Brokers deliver value via three primary roles: service aggregation; service integration; and service customization, while additional roles such as service arbitrage are also possible. By virtue of this broad definition, Gartner essentially considers any intermediation offering that adds some kind of value to a cloud service to qualify as a cloud service broker. Any provider of relevant services or technology, even with the most basic intermediation capabilities and a "simple" value proposition already qualifies as CSB.

Gartner analysts draw a useful distinction between the terms "brokerage" and "broker". They highlight the fact that these terms are often used interchangeably but actually, refer to different concepts. In Gartner's view, a broker is "a person, company or piece of technology that delivers an instance of brokerage or, the specific application of a mechanism that performs the intermediation between consumers and providers".

Forrester (Forrester ,2012), on the other hand, defines a Cloud Service Broker as the most complex business model, offering a wide value contribution in the emerging cloud space. Essentially, this model leverages skills and capabilities from all three of the traditional business models of software, consulting, and infrastructure. They argue that there are no brokers in existence yet. Only integrated or aggregated services which bring some kind of value out of the composition may qualify as a broker. According to Forrester an intermediary has to offer a certain complex "combined" value proposition in order to qualify as broker. Forrester also distinguishes three types of Cloud Brokers, according to the level of the cloud stack at which they operate, being Simple Cloud Broker, Full Infrastructure Broker, and SaaS Broker (Forrester, 2012).

The Broker@Cloud consortium defined a taxonomy of cloud service brokerage capabilities (Verginadis et al., 2013). This taxonomy represents a synthesis of present day views on cloud service brokerage, incorporating frequently-cited viewpoints by analysts such as Gartner (Plummer et al., 2011), Forrester (Forrester 2012) and NIST (NIST, 2011), and those of the

Broker@Cloud consortium. The taxonomy takes into consideration six different dimensions of brokerage capability:

i) Service Discovery
ii) Service Integration
iii) Service Aggregation
iv) Service Customisation
v) Service Quality Assurance
vi) Service Optimisation

Veloudis et al. (Veloudis et al., 2014) propose a minimal Cloud Service Brokerage (CSB) model based on the traditional Service Oriented Architecture (SOA) model by Massuthe et al. (Massuthe et al., 2005) and with a focus on software-based Cloud Services. The proposed model is minimal in the sense that it only identifies the relevant roles, without making any specific assumptions about potential interrelationships between the underlying entities that assume these roles. The CSB model extends the SOA model through the introduction of an additional role, namely the Hosting Platform, through which the various services are consumed, and the substitution of the Service Registry role with the more complex Broker role.

5.1 Optimization Factors for Service Placement

5.1.1 Application Characteristics

5.1.1.1 Mobile Application Types and Categories

Smartphones market and its applications are growing rapidly every year. As of July 2015, there were 1.6 million apps available for Android users and 1.5 million apps were available in Apple's App Store (Statista, 2015). As a result, smartphones increasingly became an essential part of human life. In our daily lives, we use mobiles to do many activities such as social networking, web browsing, emailing, video watching, and gaming. Because of the unique characteristics of smartphones, they have become powerful ultra-portable personal computers supporting not only communication but also running a variety of complex, interactive applications (Banovic, 2014). Table 1 and Table 2 show the different types and categories of mobile applications.

Table 1. Mobile Application Types [Source: Flora, Wang & Chande, 2014]

	Type	Description	Example
1	Browser Access Apps	Apps are not installed in the device and there data are not stored in the device. They can be accessed through native browser by hitting the URL of the web. Their performance is dependent on the quality of the browser.	m.yahoo.com, www.google.com.
2	Native Apps	Apps are installed in the device itself. Apps data and processing occur in the device without any need to transfer data to the server.	Notes & Reminder in iPhones.
3	Hybrid Apps (Web)	Apps are installed in the device but running the apps always requires internet connection.	Social Networking Apps (Facebook, Twitter); Instant Messengers (Skype). E-Commerce (Flipkart); Internet Speed Testing (Speedtest).
4	Hybrid Apps (Mixed)	Apps are installed in the device and the application execution may or may not require internet connection.	Medical apps; few games (offline, online)

Table 2. Mobile Application Categories [Source: Flora, Wang & Chande, 2014]

	Categories	Description
1	Communications	E-mail clients, IM clients, Social networking clients, mobile/internet browsers, News/Information clients, on device portals (Java portals)
2	Games	Puzzle/Strategy, Cards/Casino, Action/Adventure, Sports, Leisure Sports
3	Multimedia	Graphics/Image viewers, Presentation viewers, Video players, Audio players, Streaming players (Audio/Video)
4	Productivity	Calendars, Calculators, Diary, Notepad/Memo/Word Processors, Spreadsheet, Directory Services, Banking and finance, Call recording, Mobile health monitoring, Mobile advertising
5	Travel	Profile Manager, Idle screens, Screensavers, Address book, Task manager, Call manager, File manager, Mobile search
6	Utilities	Profile Manager, Idle screens, Screensavers, Address book, Task manager, Call manager, File manager, Mobile search
7	Education	Alphabet, Numerical

5.1.1.2 Characteristics of Mobile Application

Mobile applications have key characteristics that make them different from their counterparts desktop and laptop applications. There are few works on the characteristics of mobile applications in the literature, such as Salmre (2004) and Flora, Wang & Chande (2014). According to a recent survey conducted by Flora et al. (2014) involving mobile companies, mobile app development team members, mobile experts, researchers, and relevant stakeholders, mobile applications are classified into three categories: Hardware, software (application interaction, application development, and application security), and communication. Following are the key characteristics of mobile applications as identified by Salmre (2004) and the results of the survey by Flora, Wang, & Chande (2014).

Table 3. Characteristics of mobile applications [Source: Flora, Wang, & Chande, 2014; Salmre, 2004]

Category	Characteristic	Description
Hardware-related Characteristics	Less Power	Mobile app has small line code. Therefore, it requires less disk space (less memory), less computing power, less energy consumption.
	Input Mechanism	Minimum keyboard entries (Touch, pinch, and swipe).
	Star-up Time	Ability to quickly start-up s imperative for the mobile app as the user uses the mobile apps frequently.
	Physical Parameters	Illumination, noise, vibration and motion, etc.
	Device Fragmentation	Ability to run on all mobile platforms and be compatible with different devices versions and their OS.
Software-Related Characteristics	Application Interaction	User Experience, User Interface, Integration with Information sources, Integration with other Apps, Acknowledgment about the running status of the app (Action feedback), Error Notification
	Focused purpose	Mobile apps should be focused and enabling special features.
	Short-duration activates.	Using the app for short length sessions.
	Convenience	Simple, ensuring high quality content, high value and user acceptance.
	Responsiveness	Ability to remain responsive when running long operations.
	Personalization	Creating individual content and role based on personalized context or usage.

	Localization	Ability to handle sensors that respond to device movement, numerous gestures, GPS, cameras, and multiple network protocol. Possibility to provide location-based information
	Reliability/ Reachability	Mobile applications can be considered as of being "instant access". Therefore, they can be reached at any time, and any where
	Security	Encryption, expire sessions, Request validity period (automatic session outs), Prevent repeated request.
	From Factor	Ability to be used in crowded and noisy spaces, single-or two-handed operation (Salmre, 2004)
Communication - Related Characteristics	Network Connectivity	Some apps are always connected and they are affected by the network conditions (Bandwidth, latency, unstable connection, data transfer charge, battery consumption).

5.1.2 User Preferences and User Utility

In the recent years, we have seen a steady increase in the human dependency on smartphone devices in daily life. Mobile devices have emerged rapidly from a simple communication device into multifunctional information and communication device (Ferreira, Goncalves, Kostakos, Barkhuus, & Dey, 2014). With the increase in their functionality and diversity of use, smartphone devices are predicted to be the dominant future computing devices with highest user expectations for running computationally intensive applications like those of powerful desktop, notebook, or PCs (Al-Athwari & Altmann, 2013; Shiraz, Gani, Khokhar, & Buyya, 2013).

A significant amount of work has been done on smartphone to understand how users interact with their phones in the real world (Church, Ferreira, Banovic, & Lyons, 2015; Demumieux & Losquin, 2005; Do, Blom, & Gatica-Perez, 2011; Falaki et al., 2010; Ferreira et al., 2014; Froehlich, Chen, Consolvo, Harrison, & Landay, 2007; Jesdabodi & Maalej, 2015; Jones, Ferreira, Hosio, Goncalves, & Kostakos, 2015; Kang, Seo, & Hong, 2011; Kim, Ilon, & Altmann, 2013; Lim, Bentley, Kanakam, Ishikawa, & Honiden, 2015; McMillan, Morrison, Brown, Hall, & Chalmers, 2010; Ahmad Rahmati, Tossell, Shepard, Kortum, & Zhong, 2012; Abouzar Rahmati & Zhong, 2013). A typical methodology that has been used is to install a logger application on a smartphone of the user and collect actual usage data. Their common findings are that all users have a unique device usage pattern. Despite, the significant amount of research about mobile usage with different sizes of samples , and from different populations, with each study focused on different aspect of mobile usage, most researchers agree that studying mobile device usage is still very challenging (Church et al., 2015) . Another common finding is that all users have unique usage patterns. This diversity among users revealed the need of understanding the user behavior. Better understanding of user behavior helps in multiple ways in developing new

mechanisms that better match user expectations (Falaki et al., 2010; Heikkinen, Nurminen, Smura, & Hämmäinen, 2012). Technical solutions should learn and adapt user behavior to effectively improve user experience (Falaki et al., 2010). Without understanding user behavior, it is not possible to clearly understand the impact of any optimization on user experience (Shye, Scholbrock, & Memik, 2009).

In the respect of cloud computing, the rapid growth of cloud services provided from heterogeneous cloud provider, the choice of cloud service become a significant challenge. Recently, many decision making methods have been proposed to assist the process of service selection based on user preferences such as the Preference-based cLoud Service Recommender (PuLSaR), which has been developed by (Patiniotakis, Verginadis, & Mentzas, 2015) in the context of the EU FP7 project Broker@Cloud (http://www.broker-cloud.eu/). PuLSaR is a cloud consumer preference based recommender that uses a holistic multi-criteria decision making approach for offering optimization as brokerage capability. The decision making approach considers, in a unified way, quantitative and qualitative aspects. To that end the SMICloud model (Garg et al., 2013) is extended at different levels by introducing more qualitative factors. PuLSaR distinguishes between precise quantitative service attributes (e.g. service response time) expressed as crisp values and imprecise qualitative, more intuitive characteristics (e.g. service reputation) expressed as linguistic terms and fuzzy numbers. With such an approach users should be enabled to explicitly express fuzzy levels of requirements (e.g. bad, ok, good) and apply a fuzzy multi-criteria decision making method for preference-based ranking of cloud services. Specifically, it develops and implements a fuzzy Analytic Hierarchy Process (fuzzy AHP) approach that solves the problem of service ranking and allows the unified multi-objective assessment of cloud services. Thus, PuLSaR copes with the implicit uncertainty or imprecision that exists in the cloud consumer's preferences. It also lets consumers register feedback that cloud service providers can use to improve services.

5.1.3 BASMATI Optimization Requirements and Specifications

BASMATI aims at providing a set of technologies for runtime optimization of brokerage and offloading decisions of mobile applications. Considering the BASMATI cloud federation, the choice of the cloud provider who can provide the resources required to execute a particular application is a complex decision due to the heterogeneity of the services provided by the different providers in terms of costs, resources, and technology. In fact, when deciding which part of the application to offload and which cloud provider to consider, a wide range of factors affects the offloading decision according to the application requirements and the objective of optimization.

Considering the different constraints affecting the resource allocation, based on the application requirements (e.g., storage, processing), there is a need for solutions supporting the decisions on placement and offloading of specific application services. BASMATI will aim at supporting the

decisions of which components should be offloaded as well as when and where to place and/or offload new service instances generated at runtime.

The BASMATI run time optimization of brokerage and offloading decisions will provide solutions in the form of both the technologies and methods to optimize at runtime the parameters and the settings governing the objective function of the static and dynamic brokerage and offloading model according to runtime situations / contexts. To do that, BASMATI will leverage the information derived from the monitoring subsystem to infer the actual behaviour of the application in order to optimize its allocation of resources. User, application, situational-awareness and the adaptation patterns characterizing the mobile applications will be leveraged by the BASMATI brokerage platform to support both an a-priori decision making and a runtime optimization process involving application placement and offloading.

In this regard, BASMATI will provide:

i) A cost model for predicting the required resources for the application execution.
ii) An ad-hoc complete set of algorithms and methodologies aimed at identifying the different the optimal set of resources to assign to the mobile cloud applications, and
iii) An integrated set of techniques and solutions for driving multi-objective optimal offloading of applications.

In multi-objective optimization problems, two or more objective functions should be considered simultaneously. There is no unique solution for multi-objective optimization problems, but instead, a set of good trade-off solutions (Pareto optimal set) (Coello, 2005). The primary goal of multi-objective optimization is to model a decision-maker's preferences (ordering or relative importance of objectives and goals), one valid categorization of the methods should be related to the way the decision-maker articulates these preferences. So, a priori articulation of preferences implies that the user indicates the relative importance of the objective functions before running the optimization algorithm, a posteriori articulation of preferences implies the selection of a single solution from a set of mathematically equivalent solutions, while in interactive articulation of preferences the decision-maker is continually providing input throughout the execution of the algorithm (Marler & Arora, 2004).

BASMATI will deliver innovative solutions built upon existing state-of-the-art approaches (including machine-learning and economic-based approaches) able to efficiently address the resource selection problem in a complex, distributed and constrained scenario, and to determine when, what, where to offload (parts of) the mobile cloud applications. Runtime optimizations will take place to perform the fine-tuning of the applications depending on specific, unpredicted or unpredictable actually occurring conditions. The challenge here is to create an optimization mechanism that is both effective and low cost, both in terms of costs (energy, time, monetary) considering the following requirements for making the decision:

5.1.3.1 User context

The user context information such as his mobility and the remained energy level should be considered when making the offloading decision. That is, user context (e.g., mobility and energy level) have a significant effect on the offloading decision. For example, when the remained energy in the smartphone user is low and have no access to charge his mobile, in this case he would prefer to offload the heavy computation tasks to be executed on the cloud to save the battery energy. Also, in case of the user mobility, a user may lose connectivity to the internet because of changing his location but might reconnect again later after short time or might not reconnect to the internet for a long time. Therefore, the availability of the connectivity for the mobile user varies according to the current location of the user. Also, the availability of the network connectivity and its data rate for the mobile user may change during the remote execution time for one application (Magurawalage, Yang, Hu, & Zhang, 2014). In addition, disconnection during the application execution can negatively affect the user experience.

5.1.3.2 Application analysis

Application Analysis is a significant step for enabling any type of optimization and trade-off investigation. BASMATI will deliver innovative solutions built upon existing state-of-the-art approaches that have been proposed for analysing application behaviour.

In general, we can distinguish between different kinds of constraints that hinder the achievement of the BASMATI vision with regards to the application analysis:

Cloud constraints:

Such as the speed of cloud server, cloud server storage, cloud server memory, and the cost charged by the cloud service provider for the computation.

Network constraints:

When making the offloading decision it is necessary to consider the state of the available network bandwidth between the smartphone and the cloud. There are two modes of network connection between the smartphone and the cloud. Those are WiFi network and cellular network (3G or 4G). The network bandwidth capacity varies according to the type of the used network. It also varies according the current usage of the link between the smartphone and the cloud.

Mobile constraints:

It includes mobile speed, CPU load, memory, and storage. In case of the CPU, the mobile CPU can be either idle, or have utilization from 0-100%. Therefore, this constraint should be considered when making the offloading decision.

Application Constraints:

Such as data size, Computation rate, and application type. For example, some applications (e.g., chess game) have small size of data but the amount of computation is extremely large. Whereas

other applications (e.g., image retrieval) have large size of data but the amount of computation is very small (Kumar & Lu, 2010). Therefore, the application type has an influence on the offloading and need to be considered in the offloading decision.

5.1.3.3 Cost

Before the smartphone runs the application, the cost of running the application locally on a smartphone as well as the offloading cost for the executing the application on the cloud should be estimated.

The cost estimation consists of three subcomponents:

Time Cost:

It measures the time required to execute the application locally on the smartphone and the time required to execute the application on the federated cloud. The estimation of the time cost includes the following:

Local time cost: This is the time required to run the application locally on the smartphone. It is estimated value depends on the amount of the computation and the smartphone speed.

Offloading Time Cost: This is the time required for running the application on the cloud. It comprises of time required for transmitting the required data for the application, waiting for the result, and receiving execution from the federated cloud. The calculation of the time cost depends on the size of the data to be sent and received, the bandwidth capacity for sending and receiving the data, the computation rate, and the server speed of the cloud (Kumar & Lu, 2010; Xia et al., 2014).

Energy consumption cost:

To support the offloading decision, the energy consumption cost for local and offloading execution of the application need to be estimated before deciding whether to offload or not as follows:

Local energy consumption cost: This is the energy consumed by executing the computation locally on a smartphone can be calculated based on the power consumed and the average time required to run the application on the smartphone .

Offloading energy consumption cost: This cost is the energy consumed by the mobile device for executing the application on the cloud. This includes the energy consumed for transmitting the required data of the application to the cloud, waiting for the cloud to complete the execution, and receiving the result from the cloud.

Monetary Cost:

Offloading the application to the cloud requires cost for the smartphone user based on the consumed resources for communicating and running the application on the cloud. It comprises communication cost and computation cost.

Communication cost: This involves the cost of wireless/ mobile networks and the cloud. The user can offload his application to the MCC by either accessing WiFi or use 3G/4G mobile network. However, he can reduce the cost by using WiFi because wireless networks such as 3G links will charge him more cost. The communication cost is based on the amount and price of the data traffic for the communication required to execute the application on the cloud. This cost is determined by the service agreement between the user and the network provider.

Computation cost: This is the usage cost of the cloud resource. Different cloud service providers charge different rates for the same service. Usually, cloud service provider measures the computation cost based on a number of CPU cycles, storage, and communication traffic (in and out) of a cloud (Zhang, Kunjithapatham, Jeong, & Gibbs, 2011).

5.2 Optimization Methods

5.2.1 Machine Learning Applications

Many popular cloud providers already offer their customers the ability to place their application into different, geographically sparse, datacenters. It is, for example, the case of Amazon EC2 (with its Availability Zones) and Microsoft Azure. With the advent of the multi-cloud and the technologies to make Cloud platforms interoperable (Toosi et al., 2014) the problem shifted from the brokering of applications within resources controlled by the same entity to resources controlled by many Cloud providers. In the multi-cloud context, as is the case of BASMATI, it is the main task of the brokerage platforms to offer the ability to dynamically choose, on behalf of the user, the best resources for the applications.

Dynamic application placement (Li, Tordsson & Elmroth, 2011; Wang et al., 2013) considers the requirements of applications to be variable (e.g. request spikes), as well as the conditions of the cloud providers (e.g., variable prices).

In this case, the brokering approaches should run continuously in order to adapt the placement to the changing environment (Taleb and Ksentini, 2013). The dynamic brokering is similar to the static one, but it presents additional challenges, in particular, the management of monitoring, the consideration of the cost of the migration and the mechanisms to enable service availability and continuity (Rehman et al., 2015). Many brokering strategies consider data as the main entity to move across different cloud resources. Finding a proper placement for the data requires taking into account data availability and often requires dealing with other issues, such as legal and economic aspects (Berenbrink et al., 2013). However, moving data is not enough for mobile cloud applications, as also the computation shall happen closer to the users (Malet et al., 2010).

Due to the importance of the problem and the diffusion among industry and research of multi-cloud environments, dynamic brokering of computation has been tackled using many different technologies. Several of the approaches employ linear programming formulations and rule-based algorithms (Berenbrink et al., 2013; Malet et al., 2010), which offers better results in

terms of scalability but suffer when the dimension of the problem is large. These issues fostered the realizations of high-level approaches which combine the ability to work with highly dimensional problems such as solutions based on genetic algorithms (Anastasi et al., 2014), machine learning (Unuvar et al., 2015) and economic-modelling approaches (Altmann and Kashef, 2014).

Machine Learning (ML) is a wide area of computer science embracing a series of tools, methods, models and algorithms, which are designed to learn from stored data and make a prediction on new data. In the context of BASMATI, ML will be exploited to resolve problems such as classification, clustering, and ranking of application and resources. Classification requires the labelling of data pieces in order to assign them to a category. In classification, if the set of labels to be assigned to the data items are given beforehand, it is called supervised learning. Otherwise, if the labels are unknown a priori, the problem becomes a clustering problem, in which labels themselves are to be learned. In the context of dynamic application placement, ML-based classification has been exploited to assign virtual machines to data centers by labelling each datacenter in accordance with its ability to satisfy a given QoS (Unuvar et al., 2015). The ranking problem in ML is usually referred to as learning to rank (L2R) (Li 2014; Cheng et al., 2012; Hopkins & May, 2011; Liu 2009) and provides a ranking of data items according to defined objectives. A L2R-based function, which scores a set of candidate objects according to their relevance to a given query, is learned from a ground-truth composed of many training examples. The scoring function learned by a L2R algorithm aims to approximate the ideal ranking from the examples observed in the training set. L2R models are usually classified into three broad categories: point-wise, pair-wise and list-wise. Point-wise methods are regression or classification algorithms aiming at predicting the relevant label associated with each query-object pair in the training set. Pair-wise methods consider pairs of objects as training instances, and they explore scoring functions that are able to discriminate the best document among the two. The learning process for point-wise models tries to optimize loss functions such as root mean squared error (RMSE) with respect to object relevance labels, while pair-wise methods typically optimize the number of misclassified pairs. IR quality measures that are a function of the full set of results for a given query cannot be directly optimized by the above two approaches. To this end, list-wise methods have been introduced to directly optimize list-based metrics.

When it comes to selecting an ML method for solving a problem, there are many dimensions that need to be considered, such as the size of the training data, the characteristics and the number of the features, and the requirements in terms of performance. Therefore, several methods shall be evaluated. Logistic regression is the simplest form of learning and can be used for both classification and ranking. Logistic regression can be made very efficient exploiting parallelization (Chu et al., 2007), but it requires the features to be linearly dependent. In the case of non-linear features, or when the number of features is high, Support Vector Machines (SVM) (Joachims, 2012) would represent a better choice. However, SVMs may face performance

problems, as they are usually slow to train. For example tree-based learning, using random forest, can be used when the number of features is high, and recent development (Lucchese et al., 2015) improved its performance over large training sets. Two of the most effective L2R-based rankers are based on additive ensembles of regression trees, namely GRADIENT-BOOSTED REGRESSION TREES (GBRT) (Friedman, 2001), and LAMBDA-MART (λ-MART) (Burges, 2010). Due to the thousands of trees to be traversed at scoring time for each document, these rankers are also the most expensive in terms of computational time, thus impacting on response time and throughput of query processing. Therefore, devising techniques and strategies to speed-up document ranking without loosing in quality is definitely an urgent research topic (Lucchese et al., 2015).

5.2.2 Socio-Economic Models

The broker is one of the key elements for enabling federation of Clouds and auto-scaling application. Cloud brokers have different delimitations by organizations dealing with cloud paradigm (Altmann et al., 2008; Khanna & Jain, 2015). The application broker is defined as an automated entity with the following responsibilities (Grozev & Buyya, 2014) :

- Automatic provisioning and management of resources (e.g. virtual machines (VMs) and storage) for a given application across multiple clouds. Typically, this would include allocation and de-allocation of the resources.
- Deployment of the components of the application in the provisioned resources automatically.
- Scheduling and load balancing of the incoming requests to the allocated resources.

Therefore, provisioning the resources within the federated clouds could be driven by market-oriented principles for efficient resource allocation depending on user QoS targets and workload demand patterns. The commonly used economic models (Buyya et al., 2002) that can be employed for resource brokering environment include:

5.2.2.1 The Commodity Market Model

In this model, the resource providers define their price and customers define their needs accordingly. In this case, pricing policies will depend on many parameters such as usage time (e.g peak-load pricing) or usage quantity (e.g. price discrimination). The pricing schemes can be either flat (fixed), usage duration (time), subscription based, or variable price based on the supply-and-demand (Altmann, Courcoubetis & Risch, 2010; Breskovic, Altmann & Brandic, 2013; Altmann et al., 2008). For executing applications, the resource broker is responsible for carrying out the following steps (on behalf of the user):

1. The broker identifies service providers;
2. It identifies the resources which are suitable for a given application and determines their prices;

3. It selects resources that meet the user's utility function and objectives (lower cost and deadline requirements met). Typically, while selecting the resources, the broker uses heuristics and/or historical knowledge and mapping jobs to them.
4. It manages the job processing using the resource services and also it issues payments as agreed.

5.2.2.2 The Auction Model

Auctions are the most extensively market models studied through the literature. The auction model supports one-to-many negotiation, between a service provider (seller) and many consumers (buyers). The negotiation is reduced to a single value (i.e. price). The auctioneer sets the rules of the auction, acceptable for the provider and the consumers. Auctions basically use market forces to negotiate a clearing price for the service. Auctions can be classified into five types: English auction (first-price open cry); first-price sealed-bid auction; Vickrey (second-price sealed-bid) auction; Dutch auction; and double auction (continuous).

5.2.2.3 The Posted Price Model

The posted price model is similar to the commodity market model, except that resources and offer prices are advertised in order to attract (new) consumers to establish market share or motivate users to consider using cheaper slots. In this case, brokers use posted prices without any need to negotiate directly with SPs for the price, because the posted prices are generally cheaper compared to regular prices. The posted price offers will have usage conditions, but they might be attractive for some users.

5.2.2.4 The Bargaining Model

Unlike the previous models in which the broker pays access price fixed by the SP, the barraging model resource broker bargains with Service Providers (SPs) for lower access prices and higher usage durations. Both brokers and SPs negotiate with each other to meet their objectives. The SPs may start with a higher price and the brokers with a very low price. They both negotiate until they reach a mutually agreeable price or one of them is not willing to negotiate any further. This negotiation is driven by user requirements (e.g., a deadline is too relaxed) and brokers can take risks and negotiate to minimize the cost as much as possible to meet the user requirements. To avoid the lower utilization of resources, SPs might be willing to reduce the price instead of wasting resource cycles. Brokers and SPs generally employ this model when market supply and demand and service prices are not clearly established. The users can negotiate a lower price with the promise of some kind of favour or the promise of being a customer of SP's services even in the future.

5.2.2.5 The Tendering/Contract-net Model

This model is one of the most widely used models to negotiate for a service in a distributed problem-solving environment. It is based on the contracting mechanism model used by businesses to manage the exchange of services and goods. It helps in finding an appropriate service provider to work on a given task. A user/resource broker announces its requirements for

a given service is called the manager and the provider that might be able to provide the resources is called the potential contractor.

5.2.2.6 The Bid-Based Proportional Resource Sharing Model

Market-based proportional resource sharing systems are quite popular in cooperative problem-solving environments such as clusters (in a single administrative domain). In this model, users compete for shared resources in a cluster. The resources in this model are proportionally based on costs that competing users are willing to pay.

5.2.3 Multi-Objective Optimization

The concept of offloading the computation and data to the cloud is used to address the limitation of mobile devices by using the resources provided by the cloud rather than the mobile device itself to run the mobile application (Fernando, Loke & Rahayu, 2013). Computation offloading is a solution to augment the mobile's capabilities by migrating computation from mobile phones to more powerful and resourceful computing servers located in the cloud (Kumar, Liu, Lu & Bhargava, 2013).The process of moving computation from mobile device to the cloud is called computation offloading. It is typically used to boost the computational capability of a resource-constrained mobile device. In mobile Cloud computing (MCC) environment, the computation offloading is defined as the mechanism of migrating resource-intensive computation from mobile device to the resource-rich cloud (Enzai, Idawati & Tang, 2014).

Recently, several computational offloading frameworks have been proposed for offloading computational intensive mobile applications partially or entirely to the cloud. The latest developments in the computation offloading frameworks in MCC have aimed at augmenting the resources of mobile devices by leveraging the resources and services of the cloud (Ahmed, Gani, Sookhak, Ab Hamid, & Xia, 2015). Most of the existing offloading frameworks focus on what components of the application to offload, how to offload the components and where to offload the intensive component of the application (Shiraz, Sookhak, Gani & Shah, 2015).

The state of the art application offloading frameworks are designated to address different objectives. Among popular objectives are minimizing energy consumption, minimizing execution time, minimizing the monetary cost, and reducing the network latency. Table 4 presents a comparison between the state-of-the-art computation offloading frameworks based on the offloading objectives.

Table 4 Comparison of Computation offloading frameworks based on their Objectives

Offloading Objective	Literature						
	ThinkAir, (Kosta et al., 2012)	*MAUI (Cuervo et al., 2010)*	*Wolski at al., 2008*	*Cuckoo (Kemp et al., 2012)*	*CloneCloud (Chun et al., 2011)*	*Phone2Cloud (Tarkoma. et al., 2014*	*Calling the Cloud (Giurgiu et al., 2009)*
Minimizing Energy consumption	O	O		O	O	O	
Minimizing Execution time(Improve the performance)	O	O	O	O	O	O	O
Minimizing monetary cost							
Reducing Network Latency							O

ThinkAir (Kosta et al. (2012)) which already introduced in section 2.2 & section 4.3, aims at reducing the execution time and the energy consumption It logs the energy consumption, execution time, and network conditions, and use them to make a decision whether a method should be offloaded or not. However, due to the intensive profiling mechanism, the offloading decision of the ThinkAir framework is complex.

The energy- ware mobile application offloading framework MAUI Cuervo et al. (2010) which also introduced in section 2.2 continuously collects essential data such as energy consumption, CPU utilization, and network bandwidth condition, at runtime. The collected data by the profiler is then used to decide whether the method should be executed locally or remotely. Although MAUI significantly improves the battery life of a mobile device and incorporates the user mobility, it does not address the transmission latency and scalability, and does not provide the QoS features (Ahmed et al., 2015). In addition, the MAUI mechanism for run-time application profiling and solving involves additional computing resources.

Wolski et al. (2008) consider the bandwidth aspect and present a framework for making computation offloading decisions to improve the performance. They have examined various decision strategies for offloading in a grid computing case. The authors predict the time required to execute the computation locally and remotely and use offloading to minimize total execution time. They do not consider the energy aspect of offloading as they used the bandwidth between the local and remote endpoints as the bottleneck.

Cuckoo Kemp et al. (2010) aims at minimizing the energy consumption and execution time of the application by offloading mobile applications to the cloud or nearby cloud servers . The frame work architecture is already introduced in section 2.2. Although the decision parameters of the Cuckoo are decided at run time, it takes static decisions for offloading that are context unaware.

As already introduced in Section 2.2 the main goal of CloneCloud Chun et al. (2011) is to optimize the overall execution cost. The execution cost comprises energy cost and execution time. The energy consumption cost consists of CPU activity, display state, and the network state. The computation cost takes values from the clone cost variables when the method runs on the clone of the mobile device, or from the mobile device cost variables. The migration cost sums the individual migration cost of those invocations whose method have migration points.

phone2Cloud Tarkoma et al. (2014) have developed a computation offloading-based system for energy saving on smartphones called phone2cloud. The main aim of this system is to reduce the energy consumption and enhance the performance through reducing the execution time. They implemented the prototype of the phone2cloud on Android and Hadoop environment.

The objective of Calling the cloud Giurgiu et al. (2009) framework which was already introduced in section 2.2 is to minimize the interaction latency between the mobile device and the cloud server while considering the exchange data overhead. The Calling cloud framework reduces the memory consumption, communication cost, and interaction time. However, due to the dynamic analysis, profiling, synthesis, runtime partitioning and offloading, it employs compute-intensive offloading process. The framework also requires continuous synchronization that keeps the mobile device in active state for the whole session of distributed platform.

In two previous studies on offloading algorithms, bandwidth was the only network parameters considered (Kumar & Lu, 2010; Wolski et al., 2008). Furthermore, the authors assumed that the same power consumption is required for sending and receiving data, but Feeney and Nilsson (2001) showed that wireless network interfaces exhibit a complex range of consumption behavior (Feeney & Nilsson, 2001). Hence, factors such as packet size, the number of broadcasts, and point-to-point traffic need to be considered when designing energy-aware offloading protocols. Wen et al. (2012) considered the features of wireless channels and showed that execution policies depend on the input data size and completion deadline for the application, as well as the wireless transmission model (Wen, Zhang, & Luo 2012).

Although a few approaches proposed a context-aware offloading scheme for mobile cloud computing information (Ghasemi-Falavarjani, Nematbakhsh & Ghahfarokhi, 2015; Lin et al., 2013; Zhou et al., 2015a; Zhou et al., 2015b), they did not consider all of the cost factors which incurred by the offloading , the context of the user, and his preference. For example, Lin et al. (2013) proposed a context-aware decision algorithm, called CADA , to optimize the performance of the mobile device with various optimization criteria, including short response time and low

energy consumption. The proposed system profiles the user location and time of the day when tasks are offloaded for remote execution. The algorithm takes the decision whether to offload tasks or not based on historical log records such that the task is offloaded if its energy consumption when it was offloaded in the past is lower than when it was executed locally for the same time of day and the same geographic location. Such approach that depends on historical records might not be suitable for present conditions and could lead to inaccurate offloading decisions.

Zhou et al. (2015) propose a context-aware offloading decision algorithm that aims to provide code offloading decisions at runtime on selecting wireless medium and which potential cloud resource (nearby mobile cloud, cloudlet, or public cloud VMs) as the offloading location based on the device context. They present cost estimation models for each of those offloading location and use the estimation results and device context to provide offloading policies of where, when and how to offload for the mobile application. The cost estimation model includes the energy consumption and task execution time but didn't include the monetary cost.

Ghasemi-Falavarjani et al. (2015) investigated the resource allocation problem in the mobile cloud. They developed a context-aware offloading middleware for mobile cloud (OMMC) to collect contextual information of mobile devices, subtasks, and environmental variables, and also to manage the offloading process. However, regarding resource allocation, it only optimizes the energy consumption and execution time of offloading while considering some constraints such as user's acceptable deadline, service providers' residual energy , and budget constraint.

Magurawalage et al. (2014) proposed a system architecture for mobile cloud computing (MCC) that includes a cloudlet layer located between mobile devices and their cloud infrastructure or clones. This middle layer is called a cloudlet layer as it composed of cloudlets. Cloudlets are deployed next to IEEE 802.11 access points and serve as a localized service point closed to mobile devices to improve the mobile cloud services performance. They proposed an offloading algorithm on top of this architecture with the purpose of deciding whether to offload to a clone or a cloudlet. The decision-making considers the energy consumption for task execution and the network status while satisfying the response time constraints of a certain task.

Kovachev, Yu & Klamma (2012) use integer liner programming to address adaptive computation offloading as an optimization problem. Their approach considers energy usage, available memory and CPU as the criteria for offloading. The algorithm adapts dynamic approach to make the offloading decision by solving a new optimization problem each time parameters such as the available bandwidth and memory updated their values in the model. Another work by (Wu, 2013) takes into consideration network unavailability to make the offloading decision. The model uses an application partitioning algorithm, and an offloading decision module intelligently decides on whether to offload by considering the network availability for remote execution. (Ou, Yang & Hu, 2007) proposed CRoSS algorithm to selects

the best host for offloading according to the link cost. The cost of the link includes both the link failure rate and the bidirectional transmission rate.

5.3 Interaction with Adaptation Mechanisms

One of the key objectives of BASMATI project is to provide a highly dynamic and efficient support to application adaptivity. That is a fundamental requirement to realize next generation applications, targeting both current and future cyber-infrastructures, more and more consisting of a large and complex set of distributed and heterogeneous resources. Such infrastructures are expected to serve as the computational backend of a wide set of different applications and environments, e.g., IoT, Smart cities, Industry 4.0, large events with multimedia coverage, etc. Even more, in recent times, researchers and practitioners are envisioning, as a key enabling feature for next generation Cyberinfrastructures, the ability of platforms to autonomously react to different kinds of workloads by employing a distributed and differentiated set of resources. Such selection is expected to be performed by taking into account the sources generating the workload and the features characterizing the workload itself.

Due to the inherent complexity characterizing such infrastructures, the conception, the design and the development of applications targeting such computational environments is a complex and error-prone task. In fact, in such scenarios, it is not feasible and realistic to statically tailor applications with respect to a specific set of resources, as it is not possible to know in advance the set of actual resources that will be exploited to host the (subsets of) applications. As a consequence, applications (and their runtime support) need to be able to self-configure themselves to the dynamic set of resources provided by the infrastructure, either by adopting an ex-post (reactively) or ex-ante (proactively) fashion.

A fundamental requirement to properly exploit such a dynamic platform to optimize the performance of applications running on it, consists in the definition of an ad-hoc adaptation and reconfiguration support. By means of such a mechanism, it is possible for the application to restructure itself in order to exploit the most appropriated resources within the ones belonging to the cyberinfrastructure.

Adaptation of applications can be characterized by different extent, and follow different approaches. In the remaining of this section are presented a set of notable approaches presented in the literature.

Existing Approaches

OnTimeMeasure for Performance Intelligence for Future Internet applications

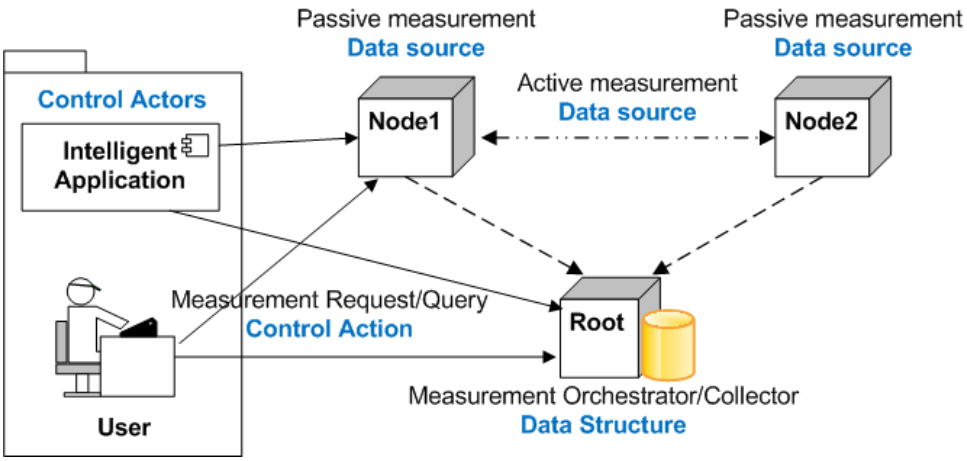

Figure 14. OnTimeMeasure [Source: Enabling Performance Intelligence for Application Adaptation in the Future Internet – *Journal of Communications and Networks, Dec. 2011*]

Autonomicity in the Future Internet applications is pretty close to the BASMATI approach, it will require a performance architecture that: (a) allows users to request and own 'slices' of geographically distributed host and network resources, (b) measures and monitors end-to-end host and network status, (c) enables analysis of the measurements within expert systems, and (d) provides performance intelligence in a timely manner for application adaptations to improve performance and scalability. Calyam et al. (2011) presented an approach, called OnTimeMeasure, that has been designed to be extensible, fault-tolerant, standards-compliant and secure (Calyam et al., 2011). It supports services to 'measure' the performance within a "user slice", 'analyze' and derive 'intelligence' that can be used in a timely manner for application adaptations to improve performance and scalability. The authors of OnTimeMeasure gave proof of the effectiveness of closed-loop orchestration used in their approach, which is critical for interoperability with other existing measurement services, particularly when a large number of application-specific measurements need to be orchestrated. They used OnTimeMeasure-enabled performance intelligence to compare utility driven resource allocation schemes in virtual desktop clouds, demonstrating how performance intelligence enables autonomic nature of FI applications to mitigate the costly resource overprovisioning and user QoE guesswork, which are common in the current Internet.

CometCloud

CometCloud (CCloud) is an autonomic computing engine for clouds and grids environments that enables the development and execution of dynamic application workflows in heterogeneous and dynamic clouds/grids infrastructures (Kim et al., 2009). It supports the on-demand bridging of public/private clouds and grids as well as autonomic cloudbursts. Conceptually, CometCloud is composed of a programming layer, service layer, and infrastructure layer. The infrastructure layer uses the Chord self-organizing overlay, and the Squid information discovery and content-based routing substrate build on top of Chord. The programming layer provides the basic

framework for application development and management including the master/worker/BOT, workflow and MapReduce/Hadoop.

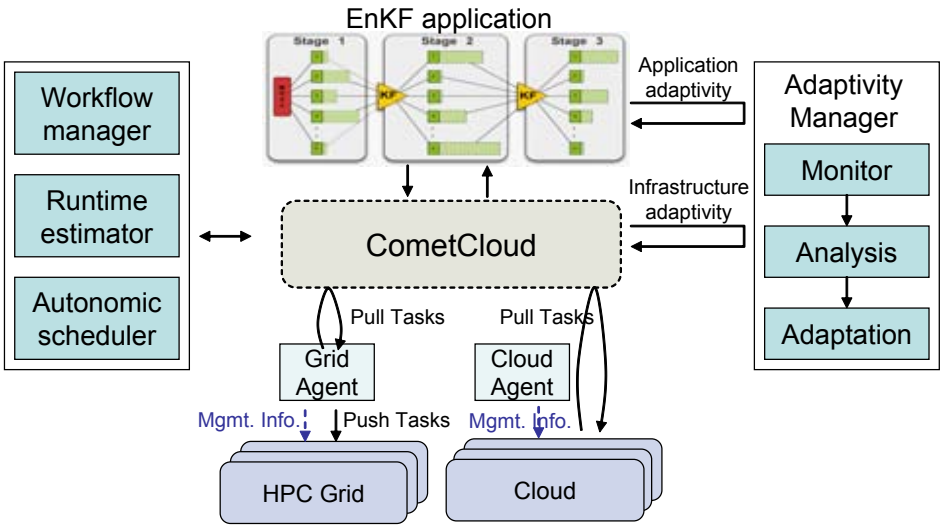

Figure 15. CometCloud [Source: Cloud Computing: Methodology, Systems, and Applications, by Lizhe Wang, Rajiv Ranjan, Jinjun Chen, Boualem Benatalla, CRC Press, 2011]

GRACE

The GRACE project (Vardhan et al., 2009) is aimed at balancing the scope and frequency of energy-saving adaptations in multiple layers by leveraging a hierarchical approach. The surrounding idea is to define two different kinds of adaptations. Expensive and infrequent global adaptation allocates resources among applications based on long-term predictions, and inexpensive per-app control seeks to make the energy-optimal use of these resources through localized short-term predictions and cross-layer adaptations. The authors of GRACE investigated an application or/and infrastructure adaptivity and determined how the adaptations affect performance as measured by time-to-completion, as well as cost. GRACE has been also integrated into CometCloud.

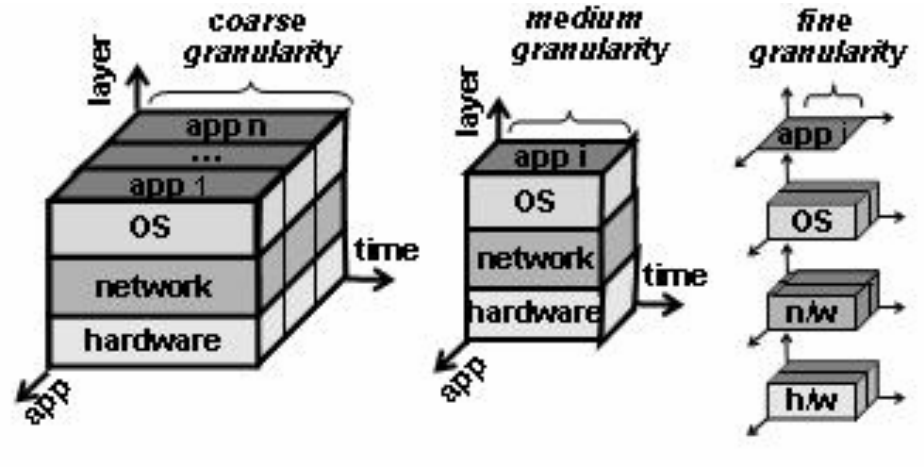

Figure 16. GRACE [Source: GRACE-2: integrating fine-grained application adaptation with global adaptation for saving energy, *Int. J. Embedded Systems, Vol. 4, No. 2, 2009*]

ThinkAir

ThinkAir (Kosta et al., 2012) proposes a different perspective with respect to the other approaches listed in this section. The idea proposed by ThinkAir is to require a limited set of modifications to an application's source code that is coupled with the ThinkAir tool-chain. The authors of ThinkAir conducted a set of experiments and evaluations with micro benchmarks and computation intensive applications to demonstrate the benefits of ThinkAir for profiling and code offloading, as well as accommodating changing computational requirements with the ability of on-demand VM resource scaling and exploiting parallelism.

Behavioural Skeletons

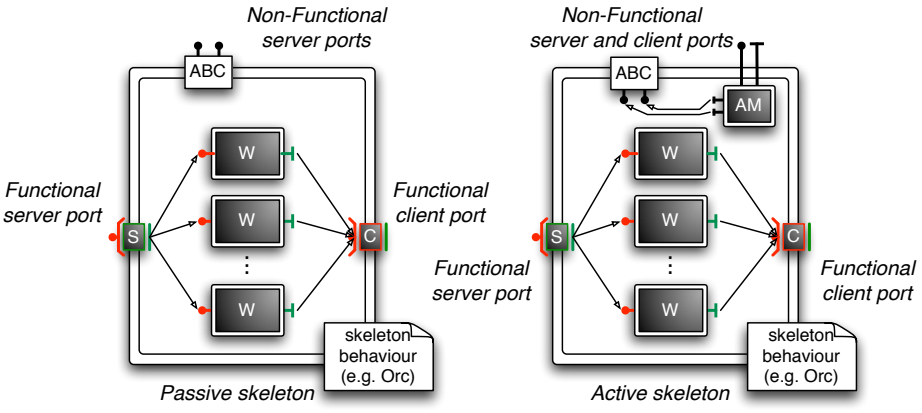

Figure 17. Behavioural skeletons [Source: Behavioural skeletons for component autonomic management on grids, *Making Grids Work*, Springer 2008]

Behavioural skeletons (Aldinucc et al., 2008) aim to abstract parametric paradigms of components' assembly, each of them specialized in solving one or more management goals

belonging to the classical AC classes, i.e. configuration, optimization, healing and protection. Behavioural skeletons represent a specialization of algorithmic skeleton concept for component management. Algorithmic skeletons have been traditionally used as a vehicle to provide efficient implementation templates of parallel paradigms. Behavioural skeletons, as algorithmic skeletons, represent patterns of parallel computations (which are expressed as graphs of components), but in addition they exploit the inherent skeleton semantics to design sound self-management schemes of parallel components. Behavioural skeletons can be identified with a composite component with no loss of generality (identifying skeletons as particular higher-order components). Since component composition is defined independently from behavioural skeletons, they do not represent the exclusive means of expressing applications, but can be freely mixed with nonskeletal components. In this setting, a behavioural skeleton is a composite component that

- Exposes a description of its functional behaviour;
- Establishes a parametric orchestration schema of inner components;
- May carry constraints that inner components are required to comply with;
- May carry a number of pre-defined plans aiming to cope with a given self-management goal.

Behavioural skeleton usage helps designers in two main ways: the application designer benefits from a library of skeletons, each of them carrying several predefined, efficient self-management strategies; and, the component/application designer is provided with a framework that helps the design of new skeletons and their implementations. The former task is achieved because (1) skeletons exhibit an explicit higher order functional semantics, which delimits the skeleton usage and definition domain; and (2) skeletons describe parametric interaction patterns and can be designed in such a way that parameters affect non-functional behaviour but are invariant for functional behaviour.

6 Requirements Analysis

The requirements generated from BASMATI use cases are presented here. A more detailed description of the use case scenarios can be found in the upcoming report D2.2 and D6.1.

6.1 Use case 1: Mobile Virtual Desktop

Mobile Virtual Desktop (MVD) is defined as a cloud service category in which the capabilities provided to the Cloud Service Customer (CSC) are the ability to build, configure, manage, store, execute and deliver users' mobile desktop functions remotely. With MVD, the user experience is achieved through a UI, which is presented through an MVD client over the network.

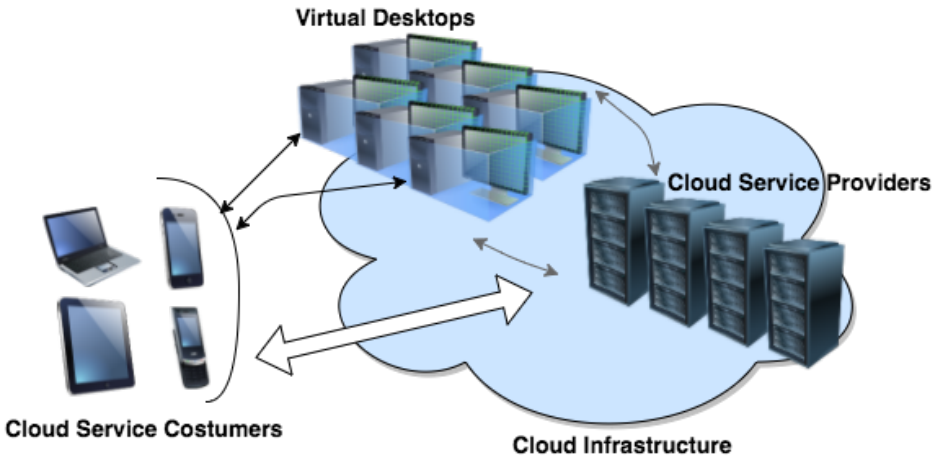

Figure 18. Conceptual view of Mobile Virtual Desktop

Instead of maintaining and running a mobile operating system and applications on CSC's devices, servers of a CSP located in the cloud are used to execute the instances of users' virtual desktops. This allows a party (e.g. an organization) to run end user's operating systems and applications, and keep their data in cloud computing environment. The following requirements derive from the existing MVD application from the observations that have been made on it.

6.1.1 User Requirements

MVD user requirements as described as follows:

- Quality of experience (QoE): MVD services need to be provided in a way that will make the experience pleasant, e.g., reduced latency, high throughput, etc. [MVD.UR.1];
- Configurability of the virtual environment: end-user should be able to specify the features characterising the hardware characteristics of the machine [MVD.UR.2];
- High availability: MVD services need to be fault tolerance and made available in regardless the actual place from which the user is accessing them [MVD.UR.3];
- Access control: only authorized user should be able to access to MVD services [MVD.UR.4].

6.1.2 System Requirements

MVD system requirements can be categorized into operational and management requirement as well as server-side requirements.

6.1.2.1 Operational and Management Requirements

Operation and management requirements include:

- **Unified management interface:** It is recommended that a CSC be capable of deploying, configuring, managing, and monitoring the MVD through a unified management interface. [MVD.SR.OMR.1]

- **Virtual desktop lifecycle management:** It is required that a CSP supports the full life cycle management of virtual desktop, including setup, test, delivery, use, maintenance, optimization, shutdown, and deletion. [MVD.SR.OMR.2]

- **Service-related resource maintenance:** It is required that a CSP maintains MVD service supporting applications and data, such as security auditing server, performance monitoring server, active directory, database, user configuration, file server, etc. [MVD.SR.OMR.3]

- **Status monitoring:** It is recommended that the current running status of virtualized resource be monitored to perform the change the status requested by a CSC. [MVD.SR.OMR.4]

- **System load monitoring:** In order to achieve an appropriate QoE, it is recommended that a CSP is capable of monitoring the system load to assign virtualized resources to a CSC [MVD.SR.OMR.5].

- **Automated management interface:** It is recommended that the MVD management solution is accessible through a consistent interface. [MVD.SR.OMR.6]

- **Accounting and charging:** It is recommended that a CSP collect accounting information based on computing power, network use, storage, memory, and/or application license fee. Accounting information is collected per service and per customer. It is also recommended that the CSP provide a charging scheme based on the accounting information and charging information transparently. [MVD.SR.OMR.7]

- **Managing and operating pre-configured environments**: It is recommended that a CSP manage and operate the pre-configured environments which are prepared after configuring the service information (such as server processing capacity, the prediction of concurrent users and the capacity of the resource etc.) and it is recommended that a CSP provide the preconfigured environment without the loss of user functionality and the degradation of performance when service is requested. [MVD.SR.OMR.8]

- **Monitoring and controlling MVD:** It is recommended to monitor and control the activities of an MVD server without impact on the performance of an MVD. [MVD.SR.OMR.9]

- **User log management**: It is recommended that a CSP keep the connection log information for all CSCs and their event logs for further security/incident analysis. [MVD.SR.OMR.10]

6.1.2.2 Server-Side Requirements

The MVD server-side functional requirements include:

- **Maintaining MVD user status**: It is required that a CSC be capable of reconnecting to a virtual desktop in the same virtual desktop state as left. [MVD.SR.SSR.1]

- **Isolation between virtual desktop functions:** It is required that the operation of the virtual desktop functions of one CSC should not be negatively impacted by the use of virtual desktop functions by other CSCs. [MVD.SR.SSR.2]

- **Graphic processing acceleration support:** In order to provide the ability to MVD clients to work with graphic intensive software packages (such as 3D computer-aided design or compression) running on the server, it is recommended that the MVD provides the acceleration of graphic processing to MVD clients. [MVD.SR.SSR.3]

- **CSC environment backup:** It is recommended that a CSP backup and restore the allocated virtual machines with user environment in order not to lose user data. It is recommended that CSP do not damage the service performance from the process backing up and restoring. [MVD.SR.SSR.4]

- **Cloud-Provider-Agnostic:** The MVD system is not locked into a cloud provider. [MVD.SR.SSR.5]

- **Transferring MVD**: It is required that a CSC is capable of reconnecting to virtual desktop regardless the real location of the MVD server. [MVD.SR.SSR.6]

6.1.3 BASMATI Platform Requirements

MVD is an application that's very sensitive to response time. Slow response time is resulting in jerky movements of the application and low satisfaction of the users. In order to maintain an acceptable QoE, response time lower than 150 ms is needed. One of the methods to reduce response time is to make MVD servers as near as possible to MVD clients. Since a single cloud service provider cannot be available in all areas, the availability of federation cloud in BASMATI is a great advantage in providing a fast response time. Because a federation cloud consisted of multiple cloud service providers can cover more areas. As a result, the traveling MVD users can access their desktops with fast response time anywhere they go.

Backend offloading of BASMATI is aiding MVD servers to balance the work load to different region and different cloud service provider. However, since the amount of data that should be transferred is big, it is advisable to schedule the transfer. For example, if a user is traveling from South Korea to Italy using an airplane. During his/her commute in an airplane, MVD servers are transferring his/her data from a server in South Korea to a server in Europe. By means of BASMATI advanced features, it will be possible to forecast the user mobility patterns, the application footprint patterns and trigger the corresponding actions aimed at transfer the application to a cloud nearby the future location of the end user, properly selected among the available ones. Application back-ends can be also restructured to match user needs while

leveraging the resources available. BASMATI platform will provide all the mechanisms for supporting these tasks.

6.2 Use Case 2: Large Events

We are focusing on investigating new applications for major events. All case studies in this context refer to the large event Das FEST in Karlsruhe (Germany). Das FEST is an open-air event held since 1985, which annually attracts 200 to 400 thousand visitors (within a time span of a weekend). The music festival is the main part, but unlike other festivals, an extensive sport, children's and family programs are also provided. In addition to sponsoring, Das FEST is mainly financed by the sale of beverages. Due to its increasing popularity, the entrance to the site had to be stopped for the first time in the season 2006.

Figure 19. The Das Fest Application

The implementation of Das FEST requires the provision of an extensive infrastructure. The continuously high number of visitor-places demands on security, which has an effect, for example, on the division of the terrain. The festival site itself is a closed area. Access to the festival site is through several entrances around the festival grounds. This makes it possible to accurately determine the number of users. The festival is easy to reach by public transport. Also, a journey by bike or on foot is possible. Parking spaces for cars and buses are available in limited numbers.

Participating in the event naturally requires physical presence on the ground. Thus the use cases cover mainly visitors who participate in the live concerts and other events of the Das FEST. However, it is getting more and more common to give a video and/or audio-streaming coverage to this kind of events, thus, as a future vision, one could imagine extending the participation to peoples who receive live streams. In this way one would provide a kind of virtual festival, which

ideally complements the festival on site with a further dimension, linking both, the real and the virtual world.

6.2.1 User Requirements

For the use cases, primarily relevant is the category of physical users[3]. The user must have a recent smartphone on which the Das FEST app is installed. For reasons of data protection, the visitor has to agree that he will be tracked. In order to create trust, the app is only supposed to perform tracking on the festival grounds and in its surroundings. The data are anonymised and it is intended to encrypt it on the server after processing.

Users should be able to receive contextualized suggestion/recommendation, based on their position, behaviour, interests. [LE.UR.1]

Users may also want to decide to which extent and way they want to be tracked (e.g., using only Bluetooth or WiFi; sharing all their past visiting paths or only the actual position) [LE.UR.2]

6.2.2 System Requirements

Supported are the operating systems Android and iOS. The app must be installed on a device with GPS capability. We assume that the device has today's usual connectivity options (Bluetooth, WLAN, a cellular network). The network coverage during the festival is bad due to the high attendance numbers on the site during the main event. Therefore, when designing the application, it must be ensured that intermittent interruptions of the online operation can occur. Certain data (such as tracking data) will be temporarily stored on the local devices until a transfer is possible.

D6.1 describes the infrastructure in detail, including problems due to bad connections. Here it is only assumed that the position of the visitor can be recorded with sufficient accuracy every 30 seconds.

6.2.3 BASMATI Platform Requirements

Front-End Application Manager and a Front-End Service: Application manager and front-end service will be used to load BASMATI services. The user interface will run on the user's mobile devices. As the computing power of modern mobile devices is sufficient, resource issues are not expected from executing the user interface. [LE.BPR.1]

Application & User Data Collector: The Application and User Data Collector is informed about the tracking activity. According to the operator's policy and the current privacy guidelines the user's position data are not to be related to any identifying personal information, not passed on to any third party and only processed for the purpose of managing the cloud, predict

[3] A detailed description of the key users and stakeholder in the use cases is presented in the D2.2.

distribution of people and queue length and determine the geo-fence for information provision. [LE.BPR.2]

Server-Side Scaling: The backend system can benefit from the BASMATI platform by means of scalability and re-location. The backend consists of different loosely coupled components [LE.BPR.3]

- Data mining process
- Search
- Data collector
- Monitoring/dashboard
- Storage

Each component can be dynamically load-balanced and scaled by the BASMATI platform during high peaks of activity. Especially the data mining process and the search need to scale horizontally because the resource consumption in these components grows in relation to the actual amount of users. Since it cannot be foreseen how many visitors will use the tracking functionality of the DAS FEST App, we will rely heavily on the scalability in this context.

Application Description: As the core components of the DAS FEST server are to be deployed and scaled differently, it is necessary to describe them in a human readable and machine interpretable way. The language to describe the application and its components should be expressive and rich enough and at the same time easy to learn and use. The use of application description templates is nice-to-have. [LE.BPR.4]

Multi-Cloud: The deployment on diverse cloud providers is relevant especially in event management: If all usage and business is performed in several days only, the availability of a service within that short period is much more relevant compared to cases where a service is used the whole year (an unavailable service for some hours is relatively unproblematic). [LE.BPR.5]

Edge Computing: BASMATI platform enables the application to infer relevant insights where they occur by allocating locally available computing resources. Like in modern data architectures a huge benefit of BASMATI is to bring the computation to the data avoiding unnecessary data movements. A local cloud storage would be nice to have for DAS FEST. It will not be introduced in 2017. [LE.BPR.6]

Deployment of Docker Containers: An additional, optional requirement within this use case is to make it possible to deploy docker container[4]. This would make the reduce necessary VM configuration effort even more compared to the deployment of jar files that typically require additional scripts to be executed before/during the deployment. Since a docker container

[4] https://www.docker.com/what-container

contains and the docker file describes all artifacts required for the deployment environment, scaling becomes even more flexible and easy to manage. [LE.BPR.7]

Monitoring: In order to enable an automatic server-side horizontal scaling, monitoring information needs to be collected within to be scaled VMs and additionally on the application level to gather as much relevant and informative. [LE.BPR.8]

- Central for all nodes belonging to a dedicated cluster
- Flexible definition of parameters on VM and application level
- Flexible Def. von Parametern auch auf App Ebene (VM)
- Response time (The DAS FEST App sends “fire and forget request” in many cases, a dedicated implementation to enable networking measure and react on the network throughput is also to be done in these cases.)
- Automatic and manual react (server start/stop/scale …)
- Monitoring the costs (optional)
  - Set budget
  - Use cost calculations for scaling decisions
  - Provider selection

Monitoring for Application Reconfiguration: As an extension to the horizontal scaling based on monitoring information, a more fine-grained scaling on application level is considered. It will be investigated to what extent it is possible to reconfigure the application, e.g. replacing a clustering algorithm calculating with lower precision but higher performance. [LE.BPR.9]

6.3 Use Case 3: TripBuilder

TripBuilder is an unsupervised system helping tourists to build their own personalized sightseeing tour.

Given a target city, the time available for the visit, and the tourist's profile, TripBuilder provides a time-budgeted tour that maximizes tourist's interests and takes into account both the time needed to enjoy the attractions and to move from one Point of Interest (PoI) to the next one.

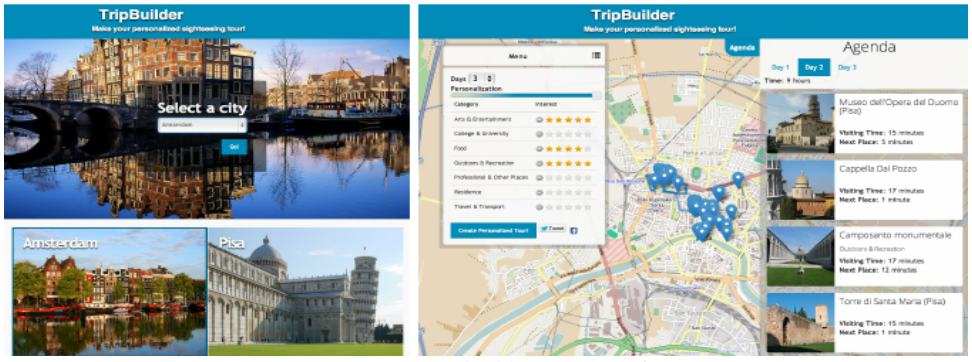

Figure 20. Tripbuilder User interface

The knowledge base feeding the sightseeing tour generation algorithm of TripBuilder is entirely mined from publicly available sources, namely, Wikipedia, Flickr and Google Maps.

TripBuilder relies on a scalable and robust Cloud architecture (combining both stream and batch processing) to download the data from the heterogeneous sources and build a huge TripBuilder knowledge base covering most popular cities worldwide.

6.3.1 User Requirements

TripBuilder users expect from the system to receive:

- Proposals of Contextualized POIs and Tripbuilding, namely to adapt the output for users depending on the actual place, time, preferences and side conditions. This is of paramount importance, as it is the core of the TripBuilder aims [TB.UR.1]
- Continuous updates to Trips, depending on what is actually happening in the area considered by the system. This allow users to be reactive with respect to the changing status of POIs, and related items, considered by the system. Such as, queues for fine art galleries [TB.UR.2]
- Answers with a very low latency. One of the items making TripBuilder a useful tool is its ability in answering queries in a timely fashion, but still taking into account the actual context in which the user is placed [TB.UR.3].

6.3.2 System Requirements

In order to satisfy the functional and nonfunctional requirements posed by TripBuilder users and the actual software, there are some requirements that are in turn posed on the system.

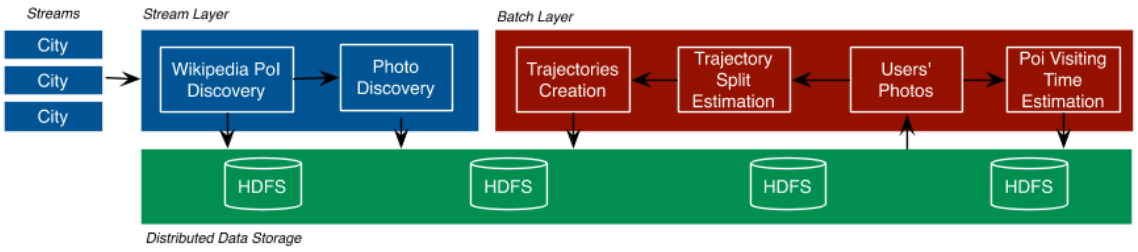

Figure 21. Components of Tripbuilder application

Access to data collections. Access to a large amount of geo- and time-referenced data in order to build the knowledge base to leverage for building Trips. Such access should be available at an acceptable cost in terms of bandwidth, latency, throughout [TB.SR.1]

Scalable processing of data. To be able to produce meaningful suggestions, taking into account the dynamicity of the system and the mass of people located in a given area, Trip Builder needs

to be able to exploit the computational resources performing data analytics in a scalable, elastic and continuous way [TB.SR.2]

Elastic Processing of end-users requests. TripBuilder needs to be elastically scaled depending on the amount of users that are accessing to the services it provides. To this end is of paramount importance to have a way to replicate, distribute, decompose the different modules composing the TripBuilder architecture, as represented in the figure reported above. Scalable and elastic process of requests/queries coming from end-users [TB.SR.3]

Distributed Computing Frameworks. TripBuilder has been conceived with scalability in mind. It has been designed to leverage state-of-the-art frameworks for distributed processing and analytics. As a consequence, to fully support, in an efficient way, the abilities provided by TripBuilder it is important to be able to support the deployment and management of distributed processing frameworks in a cloud environment. In particular, TripBuilder require the availability of Apache Spark and Storm [TB.SR.4]

Advanced data management subsystem. TripBuilder exploits the HDFS file system for enabling the data exchange between different stages composing its processing pipeline. To this end, it will need to have access to the HDFS volumes from different locations and media, leaving to the system the task of optimizing data replica as well as optimising access strategies [TB.SR.5]

6.3.3 BASMATI Platform Requirements

TripBuilder will benefit from BASMATI platform in different ways and at the different levels. As aforementioned, TripBuilder can be seen as composed of three different systems. The first two systems are parts of the application backend: one aimed at gathering and processing data, one providing the path suggestions in which the application is structured. The third system is the one on the mobile device presenting the suggestions to the user.

The systems composing the backend can benefit from BASMATI as it will provide ways and manners supporting the automatic reconfiguration, decomposition and re-location of the application.

By means of the re-location support, the application will be able to get closer to the origin of the workload, i.e., the end-users. This is a very important feature when a number of users increases as well as the mass of users moves from their home location and get concentrated in a different area (e.g., festivals, Olympic games, etc.). Furthermore, through the adaptivity support, BASMATI will allow achieving an automated scale-up and scale-down of the application accordingly with the actual requirements.

TripBuilder can benefit from its integration with BASMATI also on the mobile application side. In fact, TripBuilder could be able to have a more detailed monitoring of the application behaviour and drive, consequently, the suggestions proposed to the end-users. [TB.BPR.1]

References

Abrahamse, W., Steg, L., Vlek, C., & Rothengatter, T. (2007). The effect of tailored information, goal setting, and tailored feedback on household energy use, energy-related behaviors, and behavioral antecedents. Journal of environmental psychology, 27(4), 265-276.

Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., & Buyya, R. (2014). Cloudbased augmentation for mobile devices: motivation, taxonomies, and open challenges. Communications Surveys & Tutorials, IEEE, 16(1), 337-368.

Ahmed, E., Gani, A., Sookhak, M., Ab Hamid, S. H., & Xia, F. (2015). Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. Journal of Network and Computer Applications, 52, 52-68.

Aldinucci, M., Campa, S., Danelutto, M., Vanneschi, M., Kilpatrick, P., Dazzi, P. & Tonellotto, N. (2008, February). Behavioural skeletons in GCM: autonomic management of grid components. In 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008) (pp. 54-63). IEEE.

Al-Roomi, M., Al-Ebrahim, S., Buqrais, S., & Ahmad, I. (2013). Cloud computing pricing models: a survey. International Journal of Grid & Distributed Computing, 6(5), 93-106.

Al-Athwari, B. & Altmann, J. (2015, September). Utility-Based Smartphone Energy Consumption Optimization for Cloud-Based and On-Device Application Uses. In 12th International Conference on Economics of Grids, Clouds, Systems, and Services, Springer, LNCS.

Altmann, J., Courcoubetis, C., & Risch, M. (2010). A marketplace and its market mechanism for trading commoditized computing resources. annals of telecommunications-annales des télécommunications, 65(11-12), 653-667.

Altmann, J., Courcoubetis, C., Stamoulis, G.D., Dramitinos, M., Rayna, T., Risch, M., Bannink, C. (2008). GridEcon - A Market Place for Computing Resources. GECON 2008, Workshop on Grid Economics and Business Models, Springer LNCS, Las Palmas, Spain.

Altmann, J., Hovestadt, M., Kao, O. 2011). Business Support Service Platform for Providers in Open Cloud Computing Markets. INC2011, IEEE 7th International Conference on Networked Computing, Gumi, South-Korea.

Altmann, J., & Kashef, M. M. (2014). Cost model based service placement in federated hybrid clouds. Future Generation Computer Systems, 41, 79–90.

Amazon Web Services (2017) Amazon elastic compute cloud. Retrieved from: http://aws.amazon.com/ec2

Amazon Web Services (2017) Amazon elastic load balancing. Retrieved from: https://aws.amazon.com/elasticloadbalancing/

Amazon Web Services (2017) CloudWatch. Retrieved from: https://aws.amazon.com/cloudwatch/

Anand, A., Manikopoulos, C., Jones, Q., & Borcea, C. (2007). A quantitative analysis of power consumption for location-aware applications on smart phones. Paper presented at the 2007 IEEE International Symposium on Industrial Electronics.

Anastasi, G. F., Carlini, E., Coppola, M., & Dazzi, P. (2014, June). Qbrokage: A genetic approach for qos cloud brokering. In 2014 IEEE 7th International Conference on Cloud Computing (pp. 304-311). IEEE.

Apache Software Foundation (2017) CloudStack. Retrieved from: https://cloudstack.apache.org/

Apache Software Foundation (2016) Apache jcloud. Retrieved from: https://jclouds.apache.org.

Apache Software Foundation (2015) Apache Stratos. Retrieved from: http://stratos.apache.org.

APPRENDA (2016) Cloud Federation. Retrieved from: https://apprenda.com/library/glossary/definition-cloud-federation/.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A. & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.

Athukorala, K., Lagerspetz, E., von Kügelgen, M., Jylhä, A., Oliner, A. J., Tarkoma, S., & Jacucci, G. (2014). How carat affects user behavior: implications for mobile battery awareness applications. Paper presented at the Proceedings of the 32nd annual ACM conference on Human factors in computing systems.

Banerjee, N., Rahmati, A., Corner, M. D., Rollins, S., & Zhong, L. (2007). Users and batteries: interactions and adaptive energy management in mobile systems: Springer.

Banovic, N., Brant, C., Mankoff, J., & Dey, A. (2014). ProactiveTasks: the short of mobile device use sessions. Paper presented at the Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services.

Bardam, J.E. The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In Volume 3468 of the series Lecture Notes in Computer Science, 98-115, 2005.

Berenbrink, P., Brinkmann, A., Friedetzky, T., Meister, D., & Nagel, L. (2013, May). Distributing storage in cloud environments. In Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International (pp. 963-973). IEEE.

BEACON Project (2017). Retrieved from: http://www.beacon-project.eu/

BonFIRE (2014). European funded project bonfire-testbeds for internet of services experimentation. Retrieved from: http://www.bonfire-project.eu/.

Botta, F., Moat, H. S., & Preis, T. (2015). Quantifying crowd size with mobile phone and Twitter data. Royal Society open science, 2(5), 150162.

Bozman, J. (2010). Cloud computing: The need for portability and interoperability. IDC Analyze the Future, Sponsored by Red Hat, Inc.

Brennan, R., Walshe, B., & O'Sullivan, D. (2014). Managed semantic interoperability for federations. Journal of Network and Systems Management, 22(3), 302-330.

Breskovic, I., Altmann, J., Brandic, I. (2013). Creating Standardized Products for Electronic Markets. Future Generation Computer Systems, Elsevier, 29(4), 1000-1011.

Breskovic, I., Maurer, M., Emeakaroha, V.C., Brandic, I., Altmann, J. (2011). Towards Autonomic Market Management in Cloud Computing Infrastructures. CLOSER2011, International Conference of Cloud Computing and Service Science, Noordwijkerhout, Netherlands.

Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. Learning, 11, 23-581.

Buyya, R., Abramson, D., Giddy, J., & Stockinger, H. (2002). Economic models for resource management and scheduling in grid computing. Concurrency and computation: practice and experience, 14(13-15), 1507-1542.

Buyya, R., Ranjan, R., & Calheiros, R. N. (2010, May). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In International Conference on Algorithms and Architectures for Parallel Processing (pp. 13-31). Springer Berlin Heidelberg.

Calheiros, R. N., Masoumi, E., Ranjan, R., & Buyya, R. (2015). Workload prediction using arima model and its impact on cloud applications' qos. IEEE Transactions on Cloud Computing, 3(4), 449-458.

Calyam, P., Sridharan, M., Xu, Y., Zhu, K., Berryman, A., Patali, R., & Venkataraman, A. (2011). Enabling performance intelligence for application adaptation in the Future Internet. Journal of Communications and Networks, 13(6), 591-601.

Carlini, E., Coppola, M., Dazzi, P., Ricci, L. & Righetti, G. (2011, August). Cloud federations in contrail. In European Conference on Parallel Processing (pp. 159-168). Springer Berlin Heidelberg.

Ceilometer (n.d). Retrieved from: https://wiki.openstack.org/wiki/Ceilometer.

Celesti, A., Tusa, F., Villari, M., & Puliafito, A. (2010, July). How to enhance cloud architectures to enable cross-federation. In 2010 IEEE 3rd international conference on cloud computing (pp. 337-345). IEEE.

Cesario, E., Congedo, C., Marozzo, F., Riotta, G., Spada, A., Talia, D., & Turri, C. (2015, July). Following soccer fans from geotagged tweets at FIFA World Cup 2014. In Spatial Data Mining and Geographical Knowledge Services (ICSDM), 2015 2nd IEEE International Conference on (pp. 33-38). IEEE.

Cesario, E., Iannazzo, A. R., Marozzo, F., Morello, F., Riotta, G., Spada, A., & Trunfio, P. (2016, July). Analyzing social media data to discover mobility patterns at EXPO 2015: Methodology and results. In High Performance Computing & Simulation (HPCS), 2016 International Conference on (pp. 230-237). IEEE.

Cheng, F., Zhang, X., He, B., Luo, T., & Wang, W. (2012, November). A survey of learning to rank for real-time twitter search. In Joint International Conference on Pervasive Computing and the Networked World (pp. 150-164). Springer Berlin Heidelberg.

Chu, C., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2007). Map-reduce for machine learning on multicore. Advances in neural information processing systems, 19, 281.

Chun, B. G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011, April). Clonecloud: elastic execution between mobile device and cloud. In Proceedings of the sixth conference on Computer systems (pp. 301-314). ACM.

Chun, B.G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011). Clonecloud: elastic execution between mobile device and cloud. Paper presented at the Proceedings of the sixth conference on Computer systems.

Church, K., Ferreira, D., Banovic, N., & Lyons, K. (2015). Understanding the Challenges of Mobile Phone Usage Data. Paper presented at the Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services.

Clayman, S., Toffetti, G., Galis, A., & Chapman, C. (2012). Monitoring services in a federated cloud-the reservoir experience. Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice: Theory and Practice, 242.

Coello, C. A. C. (2005). Recent trends in evolutionary multiobjective optimization. In Evolutionary Multiobjective Optimization (pp. 7-32). Springer London.

CoherentPaaS (n.d). Retrieved from: http://coherentpaas.eu/.

collectd (n.d). The system statistics collection daemon. Retrieved from: https://collectd.org/.

CompatibleOne (n.d). Energy Efficient (Green) Cloud. Retrieved from: http://www.ens-lyon.fr/LIP/RESO/eecloud/.

Contrail (2010). Contrail-Open Computing Infrastructures for Elastic Services. Retrieved from: http://www.contrail-project.eu.

Creus, G. B., & Kuulusa, M. (2007). Optimizing mobile software with built-in power profiling Mobile Phone Programming (pp. 449-462): Springer.

Cuervo, E., Balasubramanian, A., Cho, D.k., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010). MAUI: making smartphones last longer with code offload. Paper presented at the Proceedings of the 8th international conference on Mobile systems, applications, and services.

Cuervo, E., Balasubramanian, A., Cho, D.k., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010). MAUI: making smartphones last longer with code offload. Paper presented at the Proceedings of the 8th international conference on Mobile systems, applications, and services.

Data governance (2016). Retrieved from: https://en.wikipedia.org/wiki/Data_management

Data management (2016) Retrieved from: https://en.wikipedia.org/wiki/Data_management

Datapipe (2017). Retrieved from: https://www.datapipe.com/

De Mulder, Y., Danezis, G., Batina, L., & Preneel, B. (2008, October). Identification via location-profiling in GSM networks. In Proceedings of the 7th ACM workshop on Privacy in the electronic society (pp. 23-32). ACM.

Demumieux, R., & Losquin, P. (2005). Gather customer's real usage on mobile phones. Paper presented at the Proceedings of the 7th international conference on Human computer interaction with mobile devices & services.

Dey, A. K. (2000). Providing architectural support for building context-aware applications (Doctoral dissertation, Georgia Institute of Technology).

Dey, A. K., Abowd, G. D., & Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-computer interaction, 16(2), 97-166.

Do, T. M. T., Blom, J., & Gatica-Perez, D. (2011). Smartphone usage in the wild: a large-scale analysis of applications and context. Paper presented at the Proceedings of the 13th international conference on multimodal interfaces.

Dustin, A., Bartlett, J., & Bruklis, R. (2010). Cloud Computing Use Cases White Paper. Cloud Computing Communication, 6(2), 23-29.

El Faouzi, N. E., & Klein, L. A. (2016). Data Fusion for ITS: Techniques and Research Needs. Transportation Research Procedia, 15, 495-512.

Enzai, M., Idawati, N., & Tang, M. (2014). A taxonomy of computation offloading in mobile cloud computing. Paper presented at the Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on.

F. Simini, M. C. Gonzlez, A. Maritan, and A. Barabsi. (2012a). A universal model for mobility and migration patterns. Nature 484, 7392 (2012), 96–100.

Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., & Estrin, D. (2010). Diversity in smartphone usage. Paper presented at the Proceedings of the 8th international conference on Mobile systems, applications, and services.

Fang, W., Lu, Z., Wu, J., & Cao, Z. (2012, June). RPPS: a novel resource prediction and provisioning scheme in cloud data center. In Services Computing (SCC), 2012 IEEE Ninth International Conference on (pp. 609-616). IEEE.

Feeney, L. M., & Nilsson, M. (2001). Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. Paper presented at the INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE.

Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. Future Generation Computer Systems, 29(1), 84-106.

Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. Future Generation Computer Systems, 29(1), 84-106.

Ferreira, D., Dey, A. K., & Kostakos, V. (2011). Understanding human-smartphone concerns: a study of battery life Pervasive computing (pp. 19-33): Springer.

Ferreira, D., Ferreira, E., Goncalves, J., Kostakos, V., & Dey, A. K. (2013). Revisiting human-battery interaction with an interactive battery interface. Paper presented at the Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing.

Ferreira, D., Goncalves, J., Kostakos, V., Barkhuus, L., & Dey, A. K. (2014). Contextual experience sampling of mobile application micro-usage. Paper presented at the Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services.

Ferrer, A. J., & i Montanera, E. P. (2015, May). The Role of SLAs in Building a Trusted Cloud for Europe. In IFIP International Conference on Trust Management (pp. 262-275). Springer International Publishing.

Ferrer, A. J., HernáNdez, F., Tordsson, J., Elmroth, E., Ali-Eldin, A., Zsigri, C. & Ziegler, W. (2012). OPTIMIS: A holistic approach to cloud service provisioning. Future Generation Computer Systems, 28(1), 66-77.

Ferrer, A. J., HernáNdez, F., Tordsson, J., Elmroth, E., Ali-Eldin, A., Zsigri, C., & Ziegler, W. (2012). OPTIMIS: A holistic approach to cloud service provisioning. Future Generation Computer Systems, 28(1), 66-77.

Flinn, J., Park, S., & Satyanarayanan, M. (2002). Balancing performance, energy, and quality in pervasive computing. In Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on (pp. 217-226). IEEE.

Flora, H. K., Wang, X., & Chande, S. V. (2014). An investigation on the characteristics of mobile applications: A survey study. International Journal of Information Technology and Computer Science (IJITCS), 6(11), 21.

Forrester (2012) Cloud Brokers Will Reshape The Cloud - Getting Ready For The Future Cloud Business Models. A Forrester Consulting Thought Leadership Paper.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

Froehlich, J., Chen, M. Y., Consolvo, S., Harrison, B., & Landay, J. A. (2007). MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones. Paper presented at the Proceedings of the 5th international conference on Mobile systems, applications and services.

Furletti, B., Gabrielli, L., Renso, C., & Rinzivillo, S. (2012, August). Identifying users profiles from mobile calls habits. In Proceedings of the ACM SIGKDD International Workshop on Urban Computing (pp. 17-24). ACM.

Furletti, B., Gabrielli, L., Renso, C., & Rinzivillo, S. (2013, October). Analysis of GSM calls data for understanding user mobility behavior. In Big Data, 2013 IEEE International Conference on (pp. 550-555). IEEE.

Garg, S., K., Versteeg, S., Buyya, R. (2013). A framework for ranking of cloud computing services. Future Generation Computer Systems, Elsevier, Vol. 29, pp. 1012–1023.

Gebregiorgis, S., & Altmann, J. (2015). IT Service Platforms: Their Value Creation Model and the Impact of their Level of Openness on their Adoption. Procedia Computer Science, V.68, pp 173-187.

Georgiev, P., Noulas, A., & Mascolo, C. (2014). The call of the crowd: Event participation in location-based social services. arXiv preprint arXiv:1403.7657.

Ghasemi-Falavarjani, S., Nematbakhsh, M., & Ghahfarokhi, B. S. (2015). Context-aware multi-objective resource allocation in mobile cloud. Computers & Electrical Engineering, 44, 218-240.

Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F., Renso, C., Rinzivillo, S., & Trasarti, R. (2011). Unveiling the complexity of human mobility by querying and mining massive trajectory data. The VLDB Journal-The International Journal on Very Large Data Bases, 20(5), 695-719.

Girdhar, R., Fouhey, D. F., Rodriguez, M., & Gupta, A. (2016). Learning a Predictable and Generative Vector Representation for Objects. arXiv preprint arXiv:1603.08637.

Giurgiu, I., Riva, O., Juric, D., Krivulev, I., & Alonso, G. (2009). Calling the cloud: enabling mobile phones as interfaces to cloud applications Middleware 2009 (pp. 83-102): Springer.

Goiri, Í., Guitart, J., & Torres, J. (2012). Economic model of a Cloud provider operating in a federated Cloud. Information Systems Frontiers, 14(4), 827-843.

Gonzalez, M. C., Hidalgo, C. A., & Barabasi, A. L. (2008). Understanding individual human mobility patterns. Nature, 453(7196), 779-782.

Google Data Centers. (2016). Data center locations. Retrieved from: https://www.google.com/about/datacenters/inside/locations/index.html

Google App Engine (2017) Google. Retrieved from: http://cloude.google.com/appengine/

Grozev, N. & Buyya, R. (2014). Inter-Cloud architectures and application brokering: taxonomy and survey. Software: Practice and Experience, 44(3), 369-390.

Gu, B., Sheng, V. S., Wang, Z., Ho, D., Osman, S., & Li, S. (2015). Incremental learning for v-support vector regression. Neural Networks, 67, 140-150.

Gui, F., Adjouadi, M., & Rishe, N. (2009, May). A contextualized and personalized approach for mobile search. In Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on (pp. 966-971). IEEE.

Gupta, A., & Mohapatra, P. (2007). Energy consumption and conservation in wifi based phones: A measurement-based study. Paper presented at the 2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks.

Haile, N., & Altmann, J. (2017). Evaluating investments in portability and interoperability between software service platforms, Future Generation Computer Systems.

Haile, N., & Altmann, J. (2016). Value creation in software service platforms. Future Generation Computer Systems, 55, 495-509.

Haile, N., & Altmann, J. (2015a). Structural Analysis of Value Creation in Software Service Platforms. Electronic Markets, 26(2), 129-142.

Haile, N., & Altmann, J. (2015b). Risk-benefit-mediated impact of determinants on the adoption of cloud federation. In 19th Pacific Asia Conf on Information Systems.

Hamsanandhini, S., & Mohana, R. S. (2015, January). Maximizing the revenue with client classification in Cloud Computing market. In Computer Communication and Informatics (ICCCI), 2015 International Conference on (pp. 1-7). IEEE.

Harnik, Ron (2016, February). How Can an Enterprise Benefit from a Cloud Management Platform [Blog]. Retrieved from: http://www.scalr.com/blog/how-can-an-enterprise-benefit-from-a-cloud-management-platform.

Heikkinen, M. V., Nurminen, J. K., Smura, T., & Hämmäinen, H. (2012). Energy efficiency of mobile handsets: Measuring user attitudes and behavior. Telematics and Informatics, 29(4), 387-399.

Hopkins, M., & May, J. (2011, July). Tuning as ranking. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 1352-1362). Association for Computational Linguistics.

Huerta-Canepa, G., & Lee, D. (2010, June). A virtual cloud computing provider for mobile devices. In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (p. 6). ACM.

Jesdabodi, C., & Maalej, W. (2015). Understanding usage states on mobile devices. Paper presented at the Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing.

Joachims, T. (2002). Learning to classify text using support vector machines: Methods, theory and algorithms. Kluwer Academic Publishers.

Jones, S. L., Ferreira, D., Hosio, S., Goncalves, J., & Kostakos, V. (2015). Revisitation analysis of smartphone app use. Paper presented at the Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing.

Kang, J.M., Park, C.K., Seo, S.S., Choi, M.J., & Hong, J. W.K. (2008). User-centric prediction for battery lifetime of mobile devices Challenges for Next Generation Network Operations and Service Management (pp. 531-534): Springer.

Kang, J.-M., Seo, S.-s., & Hong, J. W.K. (2011). Usage pattern analysis of smartphones. Paper presented at the Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific.

Karamshuk, D., Boldrini, C., Conti, M., & Passarella, A. (2011). Human mobility models for opportunistic networks. Communications Magazine, IEEE, 49(12), 157-165.

Kashef, M.M., Altmann, J. (2011). A Cost Model for Hybrid Clouds. GECON2011, 8th International Workshop on Economics of Grids, Clouds, Systems, and Services, Springer LNCS, Paphos, Cyprus.

Kashef, M. M., Uzbekov, A., Altmann, J., & Hovestadt, M. (2013, May). Comparison of two yield management strategies for cloud service providers. In International Conference on Grid and Pervasive Computing (pp. 170-180). Springer Berlin Heidelberg.

Kemp, R., Palmer, N., Kielmann, T., & Bal, H. (2010, October). Cuckoo: a computation offloading framework for smartphones. In International Conference on Mobile Computing, Applications, and Services (pp. 59-79). Springer Berlin Heidelberg.

Khanna, P., & Jain, S. (2015). Cloud Broker Definition Differential: Gartner Versus NIST and New Models

Kim, H., Parashar, M., Foran, D. J., & Yang, L. (2009, October). Investigating the use of autonomic cloudbursts for high-throughput medical image registration. In 2009 10th IEEE/ACM International Conference on Grid Computing (pp. 34-41). IEEE.

Kim, J., Ilon, L., & Altmann, J. (2013). Adapting smartphones as learning technology in a Korean university. Journal of Integrated Design and Process Science, 17(1), 5-16.

Kim, K., Kang, S., & Altmann, J. (2014, September). Cloud Goliath versus a federation of cloud Davids. In International Conference on Grid Economics and Business Models (pp. 55-66). Springer International Publishing.

Kim, K., & Altmann, J. (2017). Effect of homophily on network formation. Communications in Nonlinear Science and Numerical Simulation, 44, 482-494.

Klymash, M., Beshley, M., Strykhalyuk, B., & Maksymyuk, T. (2014, May). Research and development the methods of quality of service provision in Mobile Cloud systems. In

Communications and Networking (BlackSeaCom), 2014 IEEE International Black Sea Conference on (pp. 160-164). IEEE.

Kolev, B., Valduriez, P., Bondiombouy, C., Jiménez-Peris, R., Pau, R., & Pereira, J. (2015). CloudMdsQL: Querying heterogeneous cloud data stores with a common language. Distributed and Parallel Databases, 1-41.

Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012, March). Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In INFOCOM, 2012 Proceedings IEEE (pp. 945-953). IEEE.

Kousiouris, G., Kyriazis, D., Gogouvitis, S., Menychtas, A., Konstanteli, K., & Varvarigou, T. (2011, June). Translation of application-level terms to resource-level attributes across the Cloud stack layers. In Computers and Communications (ISCC), 2011 IEEE Symposium on (pp. 153-160). IEEE.

Kousiouris, G., Menychtas, A., Kyriazis, D., Gogouvitis, S., & Varvarigou, T. (2014). Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in Cloud platforms. Future Generation Computer Systems, 32, 27-40.

Kousiouris, G., Vafiadis, G., & Varvarigou, T. (2013, October). Enabling proactive data management in virtualized hadoop clusters based on predicted data activity patterns. In P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on (pp. 1-8). IEEE.

Kovachev, D., Yu, T., & Klamma, R. (2012). Adaptive computation offloading from mobile devices into the cloud. Paper presented at the Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on.

Kujala, R., Aledavood, T., & Saramäki, J. (2016). Estimation and monitoring of city-to-city travel times using call detail records. EPJ Data Science, 5(1), 1.

Kumar, K., & Lu, Y.H. (2010). Cloud computing for mobile users: Can offloading computation save energy? Computer (4), 51-56.

Kumar, K., Liu, J., Lu, Y.-H., & Bhargava, B. (2013). A survey of computation offloading for mobile systems. Mobile Networks and Applications, 18(1), 129-140.

Kurze, T., Klems, M., Bermbach, D., Lenk, A., Tai, S., & Kunze, M. (2011). Cloud federation. CLOUD COMPUTING, 2011, 32–38. JOUR.

Kyriazis, D. (2013). Cloud computing service level agreements: exploitation of research results. European Commission Directorate General Communications Networks Content and Technology Unit, Tech. Rep, 5.

Li, H. (2014). Learning to rank for information retrieval and natural language processing. Synthesis Lectures on Human Language Technologies, 7(3), 1-121.

Li, W., Tordsson, J., & Elmroth, E. (2011, November). Modeling for dynamic cloud scheduling via migration of virtual machines. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on (pp. 163-171). IEEE.

Libvirt project (2017) libvirt Virtualization API. Retrieved from: https://libvirt.org/

Lim, S. L., Bentley, P. J., Kanakam, N., Ishikawa, F., & Honiden, S. (2015). Investigating country differences in mobile app user behavior and challenges for software engineering. Software Engineering, IEEE Transactions on, 41(1), 40-64.

Lin, T.Y., Lin, T.A., Hsu, C.H., & King, C.T. (2013). Context-aware decision engine for mobile cloud offloading. Paper presented at the Wireless Communications and Networking Conference Workshops (WCNCW), 2013 IEEE.

Liu, T. Y. (2009). Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3), 225-331.

López, L., Nieto, F. J., Velivassaki, T. H., Kosta, S., Hong, C. H., Montella, R., & Fernández, C. (2016). Heterogeneous Secure Multi-level Remote Acceleration Service for Low-power Integrated Systems and Devices. Procedia Computer Science, 97, 118-121.

Lou, H. D., Li, W. G., Hou, Y. E., Yao, Q. H., Ye, G. Q., & Wan, H. (2015). Feature selection tracking algorithm based on sparse representation. Mathematical Problems in Engineering, 2015.

Lu, X., Wetter, E., Bharti, N., Tatem, A. J., & Bengtsson, L. (2013). Approaching the limit of predictability in human mobility. Scientific reports, 3.

Lucchese, C., Nardini, F. M., Orlando, S., Perego, R., Tonellotto, N., & Venturini, R. (2015, August). Quickscorer: A fast algorithm to rank documents with additive ensembles of regression trees. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 73-82). ACM.

Ludwig, H., Keller, A., Dan, A., King, R. P., & Franck, R. (2003). Web service level agreement (WSLA) language specification. IBM Corporation, 815-824.

Ma, R. K., Lam, K. T., & Wang, C. L. (2011, December). exCloud: Transparent runtime support for scaling mobile applications in cloud. In Cloud and Service Computing (CSC), 2011 International Conference on (pp. 103-110). IEEE.

Macedo, A. Q., Marinho, L. B., & Santos, R. L. (2015, September). Context-aware event recommendation in event-based social networks. In Proceedings of the 9th ACM Conference on Recommender Systems (pp. 123-130). ACM.

Machado, G. S., Hausheer, D., & Stiller, B. (2009, October). Considerations on the Interoperability of and between Cloud Computing Standards. In 27th Open Grid Forum (OGF27), G2C-Net Workshop: From Grid to Cloud Networks.

Magurawalage, C. M. S., Yang, K., Hu, L., & Zhang, J. (2014). Energy-efficient and network-aware offloading algorithm for mobile cloud computing. Computer Networks, 74, 22-33.

Majhi, S. K., & Bera, P. (2014, December). VM migration auction: Business oriented federation of cloud providers for scaling of application services. In Parallel, Distributed and Grid Computing (PDGC), 2014 International Conference on (pp. 196-201). IEEE.

Malet, B., & Pietzuch, P. (2010, November). Resource allocation across multiple cloud data centres. In Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science (p. 5). ACM.

March, V., Gu, Y., Leonardi, E., Goh, G., Kirchberg, M., & Lee, B. S. (2011). µcloud: towards a new paradigm of rich mobile applications. Procedia Computer Science, 5, 618-624.

Marinelli, E. E. (2009). Hyrax: cloud computing on mobile devices using MapReduce (No. CMU-CS-09-164). Carnegie-mellon univ Pittsburgh PA School of computer science.

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization, 26(6), 369-395.

Marosi, A., Kecskemeti, G., Kertesz, A. & Kacsuk, P. (2011). FCM: an architecture for integrating IaaS cloud systems.

Massie, M. L., Chun, B. N., & Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing, 30(7), 817-840.

Massuthe, P., Reisig, W., & Schmidt, K. (2005). An operating guideline approach to the SOA.

Masucci, A. P., Serras, J., Johansson, A., & Batty, M. (2013). Gravity versus radiation models: On the importance of scale and heterogeneity in commuting flows. Physical Review E, 88(2), 022812.

McMillan, D., Morrison, A., Brown, O., Hall, M., & Chalmers, M. (2010). Further into the wild: Running worldwide trials of mobile systems Pervasive Computing (pp. 210-227): Springer.

Monreale, A., Pinelli, F., Trasarti, R., & Giannotti, F. (2009, June). Wherenext: a location predictor on trajectory pattern mining. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 637-646). ACM.

Moreno-Vozmediano, R., Montero, R. S., & Llorente, I. M. (2012). Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. Computer, 45(12), 65-72.

Nagel, K., Kidd, C. D., O'Connell, T., Dey, A., & Abowd, G. D. (2001, September). The family intercom: Developing a context-aware audio communication system. In International Conference on Ubiquitous Computing (pp. 176-183). Springer Berlin Heidelberg.

Nagios (2009) IT Infrastructure Monitoring. Retrieved from: http://www.nagios.org

Navarro, Ted (2015) What is the Relationship Between Hybrid Clouds And Federated Clouds? Retrieved from: https://www.computenext.com/blog/what-is-the-relationship-between-hybrid-clouds-and-federated-clouds/

Neary, D. & Mark, J. (2016). ManageIQ Overview at Management and Orchestration Developer Meet-up (MODM). Retrieved from: http://www.slideshare.net/JeromeMarc2/manageiq-overview-at-management-and-orchestration-developer-modm-meetup

Newson, P., & Krumm, J. (2009, November). Hidden Markov map matching through noise and sparseness. In Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems (pp. 336-343). ACM.

NIST (2011). Cloud Compouting Reference Architecture, National Institute of Standrds and Technology, Special Publication 500-292.

OCCI & CompatibleOne (2012). Open Cloud Computing Interface. Retrieved from: http://occi-wg.org/2012/07/15/occi-compatibleone/

OGF (n.d) Open Grid Forum. Retrieved from: https://www.ogf.org/ogf/doku.php

Oliner, A. J., Iyer, A. P., Stoica, I., Lagerspetz, E., & Tarkoma, S. (2013). Carat: Collaborative energy diagnosis for mobile devices. Paper presented at the Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems.

Oliver, E. (2010). The challenges in large-scale smartphone user studies. Paper presented at the Proceedings of the 2nd ACM International Workshop on Hot Topics in Planet-scale Measurement.

Oliver, E. A., & Keshav, S. (2011). An empirical approach to smartphone energy level prediction. Paper presented at the Proceedings of the 13th international conference on Ubiquitous computing.

OPTIMIS (2013) Retrieved from: http://optimistoolkit.com

OpenNebula Project (2017) OpenNebula. Retrieved from: https://opennebula.org/

OpenTSDB (2017)The Scalable Time Series Database.
Retrieved from: http://opentsdb.net/index.html

Ortiz, S. (2011). The problem with cloud-computing standardization. Computer, 44(7), 13-16.

OSGi Alliance (2007). Osgi service platform, core specification, release 4, version 4.1. OSGi Specification.

Othman, M., Madani, S. A., & Khan, S. U. (2014). A survey of mobile cloud computing application models. IEEE Communications Surveys & Tutorials, 16(1), 393-413.

Ou, S., Yang, K., & Hu, L. (2007). Cross: a combined routing and surrogate selection algorithm for pervasive service offloading in mobile ad hoc environments. Paper presented at the Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE.

Pasricha, S., Donohoo, B. K., & Ohlsen, C. (2015). A middleware framework for application-aware and user-specific energy optimization in smart mobile devices. Pervasive and Mobile Computing, 20, 47-63.

Pathak, A., Hu, Y. C., & Zhang, M. (2012). Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. Paper presented at the Proceedings of the 7th ACM european conference on Computer Systems.

Patiniotakis, I., Verginadis, Y., & Mentzas, G. (2014). Preference-based cloud service recommendation as a brokerage service. In Proceedings of the 2nd International Workshop on CrossCloud Systems (p. 5). ACM.

Pecher, P., Hunter, M., & Fujimoto, R. (2014, December). Past and future trees: structures for predicting vehicle trajectories in real-time. In Proceedings of the Winter Simulation Conference 2014 (pp. 2884-2895). IEEE.

Pecher, P., Hunter, M., & Fujimoto, R. (2016, May). Data-Driven Vehicle Trajectory Prediction. In Proceedings of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation (pp. 13-22). ACM.

Peltonen, E., Lagerspetz, E., Nurmi, P., & Tarkoma, S. (2016). Where Has My Battery Gone?: A Novel Crowdsourced Solution for Characterizing Energy Consumption. Pervasive Computing, IEEE, 15(1), 6-9.

Petri, I., Beach, T., Zou, M., Montes, J. D., Rana, O., & Parashar, M. (2014, March). Exploring models and mechanisms for exchanging resources in a federated cloud. In Cloud Engineering (IC2E), 2014 IEEE International Conference on (pp. 215-224). IEEE.

Plummer, D. C., Lheureux, B. J., & Karamouzis, F. (2010). Defining cloud services brokerage: taking intermediation to the next level. Gartner Research Note, (G00206187).

Plummer, D. C., Lheureux, B. J., Cantara, M., & Bova, T. (2011). Cloud Services Brokerage is dominated by Three Primary Roles. Gartner Research Note G, 226509.

Prevost, J. J., Nagothu, K., Kelley, B., & Jamshidi, M. (2011, June). Prediction of cloud data center networks loads using stochastic and neural models. In System of Systems Engineering (SoSE), 2011 6th International Conference on (pp. 276-281). IEEE.

Project Jellyfish (2016) Project Jellyfish. Retrieved from: http://projectjellyfish.org/

Pu, L., Chen, X., Xu, J., & Fu, X. (2016). D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration. IEEE Journal on Selected Areas in Communications, 34(12), 3887-3901.

Quan, D.M. & Altmann, J. (2009). Resource Allocation Algorithm for the Light Communication Grid-Based Workflows within an SLA Context. International Journal of Parallel, Emergent and Distributed Systems, 24(1).

Quercia, D., Lathia, N., Calabrese, F., Di Lorenzo, G., & Crowcroft, J. (2010, December). Recommending social events from mobile phone location data. In 2010 IEEE International Conference on Data Mining (pp. 971-976). IEEE.

Qureshi, S. S., Ahmad, T., & Rafique, K. (2011, September). Mobile cloud computing as future for mobile applications-Implementation methods and challenging issues. In 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (pp. 467-471). IEEE.

Rahmati, A., & Zhong, L. (2013). Studying smartphone usage: Lessons from a four-month field study. Mobile Computing, IEEE Transactions on, 12(7), 1417-1427.

Rahmati, A., Qian, A., & Zhong, L. (2007, September). Understanding human-battery interaction on mobile phones. In Proceedings of the 9th international conference on Human computer interaction with mobile devices and services (pp. 265-272). ACM.

Rahmati, A., Tossell, C., Shepard, C., Kortum, P., & Zhong, L. (2012). Exploring iPhone usage: the influence of socioeconomic differences on smartphone adoption, usage and usability. Paper presented at the Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services.

Rai, B. K. and Srivastava, Dr. A. K. Pseudonymization Techniques for Providing Privacy and Security in HER. In International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Vol. 5, Issue 4, July - August 2016

Ranjan, R., Benatallah, B., Dustdar, S., & Papazoglou, M. P. (2015). Cloud resource orchestration programming: overview, issues, and directions. IEEE Internet Computing, 19(5), 46-56.

Ravi, N., Scott, J., Han, L., & Iftode, L. (2008). Context-aware battery management for mobile phones. Paper presented at the Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on.

Rehman, Z., Hussain, O. K., Chang, E., & Dillon, T. (2015). Decision-making framework for user-based inter-cloud service migration. Electronic Commerce Research and Applications, 14(6), 523-531.

Renso, C., Spaccapietra, S., & Zimányi, E. (2013). Mobility Data. Cambridge University Press.

Rezaei, R., Chiew, T. K., Lee, S. P., & Aliee, Z. S. (2014). A semantic interoperability framework for software as a service systems in cloud computing environments. Expert Systems with Applications, 41(13), 5751-5770.

Risch, M. & Altmann, J. (2009a). Enabling Open Cloud Markets Through WS-Agreement Extensions. Service Level Agreements in Grids Workshop, in conjunction with GRID2009, CoreGRID Springer Series, Banff, Canada.

Risch, M. & Altmann, J. (2009b) Capacity Planning in Economic Grid Markets," GPC 2009, 4th International Conference on Grid and Pervasive Computing, Lecture Notes in Computer Science (LNCS) 5529, Geneva, Switzerland, May 2009.

Risch, M., Brandic, I., Altmann, J. (2009). Using SLA Mapping to Increase Market Liquidity. NFPSLAM-SOC 2009, 3rd Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing, in conjunction with ICSOC 2009, 7th International Conference on Service-Oriented Computing, Springer LNCS 6275, Stockholm, Sweden, November 2009.

Rodero-Merino, L., Vaquero, L. M., Gil, V., Galán, F., Fontán, J., Montero, R. S., & Llorente, I. M. (2010). From infrastructure delivery to service management in clouds. Future Generation Computer Systems, 26(8), 1226-1240.

Roy, N., Dubey, A., & Gokhale, A. (2011). Efficient autoscaling in the cloud using predictive models for workload forecasting. In Cloud Computing (CLOUD), 2011 IEEE International Conference on (pp. 500-507).

Salmre, I. (2004). Characteristics of Mobile Applications. Retrieved from: http://ptgmedia.pearsoncmg.com/images/0321269314/samplechapter/salmre_ch02.pdf

Samimi, P., Teimouri, Y., & Mukhtar, M. (2014). A combinatorial double auction resource allocation model in cloud computing. Information Sciences.

Saripalli, P., Kiran, G. V. R., Shankar, R. R., Narware, H., & Bindal, N. (2011, December). Load prediction and hot spot detection models for autonomic cloud computing. In Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on (pp. 397-402). IEEE.

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4), 14-23.

Scalr (2016) Enterprise-Grade Cloud Management Platform. Retrieved from: http://www.scalr.com/

Schilit, B., Adams, N., & Want, R. (1994, December). Context-aware computing applications. In Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on (pp. 85-90). IEEE.

Schneider, C. M., Belik, V., Couronné, T., Smoreda, Z., & González, M. C. (2013). Unravelling daily human mobility motifs. Journal of The Royal Society Interface, 10(84), 20130246.

Schork, S. Kontext-sensitive Berechtigungen für Cloud-Plattformen zur Erhöhung der Datensicherheit. Masterthesis an der Fakultät IWI der Hochschule Karlsruhe, 2016

SeaClouds (n.d). D4.4 Dynamic QoS verification and SLA management approach. Retrieved from: http://www.seaclouds-project.eu/sites/default/files/seaclouds/public/content-files/deliverables/SEACLOUDS-D4.4-Dynamic_QoS_verification_and_SLA_management_approach.pdf

Shiraz, M., Gani, A., Khokhar, R. H., & Buyya, R. (2013). A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. Communications Surveys & Tutorials, IEEE, 15(3), 1294-1313.

Shiraz, M., Sookhak, M., Gani, A., & Shah, S. A. A. (2015). A Study on the Critical Analysis of Computational Offloading Frameworks for Mobile Cloud Computing. Journal of Network and Computer Applications, 47, 47-60.

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010, May). The hadoop distributed file system. In 2010 IEEE 26th symposium on mass storage systems and technologies (MSST) (pp. 1-10). IEEE.

Shye, A., Scholbrock, B., & Memik, G. (2009). Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. Paper presented at the Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture.

Silva, G. C., Rose, L. M., & Calinescu, R. (2013, December). Towards a model-driven solution to the vendor lock-in problem in cloud computing. In Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on (Vol. 1, pp. 711-716). IEEE.

Simmons, R., Browning, B., Zhang, Y., & Sadekar, V. (2006, September). Learning to predict driver route and destination intent. In 2006 IEEE Intelligent Transportation Systems Conference (pp. 127-132). IEEE.

SLA@SOI (n.d) SLA@SOI. Retrieved from: http://sla-at-soi.eu/

Smith, Math (2013) Hybrid Cloud vs Federated Cloud. Retrieved from: http://stackoverflow.com/questions/12988988/hybrid-cloud-vs-federated-cloud/14568002#14568002

Song, C., Koren, T., Wang, P., & Barabási, A. L. (2010). Modelling the scaling properties of human mobility. Nature Physics, 6(10), 818-823.

Statista (2015). Number of apps available in leading app stores as of July 2015. Retrieved from: from http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

SUNFISH Project (2017). Retrieved from: http://www.sunfishproject.eu/

Taleb, T., & Ksentini, A. (2013). Follow me cloud: interworking federated clouds and distributed mobile networks. IEEE Network, 27(5), 12-19.

Taravat, A., Del Frate, F., Cornaro, C., & Vergari, S. (2015). Neural networks and support vector machine algorithms for automatic cloud classification of whole-sky ground-based images. IEEE Geoscience and remote sensing letters, 12(3), 666-670.

Tarkoma, S., Siekkinen, M., Lagerspetz, E., & Xiao, Y. (2014). Smartphone energy consumption: modeling and optimization: Cambridge University Press.

TechTarget (2011) Federated Cloud (Cloud Federation). Retrieved from: http://whatis.techtarget.com/definition/federated-cloud-cloud-federation

Toosi, A. N., Calheiros, R. N., & Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. ACM Computing Surveys (CSUR), 47(1), 7.

Universität Berkeley (2006) CTK - Context Toolkit. Retrieved from: http://www.cs.cmu.edu/~anind/context.html#sampleapplication.

Unuvar, M., Tosi, S., Doganata, Y. N., Steinder, M. G., & Tantawi, A. N. (2015). Selecting Optimum Cloud Availability Zones by Learning User Satisfaction Levels. IEEE Transactions on Services Computing, 8(2), 199-211.

Vallerio, K. S., Zhong, L., & Jha, N. K. (2006). Energy-efficient graphical user interface design. IEEE Transactions on Mobile Computing, 5(7), 846-859.

Vallina-Rodriguez, N., Hui, P., Crowcroft, J., & Rice, A. (2010). Exhausting battery statistics: understanding the energy demands on mobile handsets. Paper presented at the Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds.

Vardhan, V., Yuan, W., Harris, A. F., Adve, S. V., Kravets, R., Nahrstedt, K. & Jones, D. (2009). GRACE-2: integrating fine-grained application adaptation with global adaptation for saving energy. international Journal of embedded Systems, 4(2), 152-169.

Veloudis, S., Friesen, A., Paraskakis, I., Verginadis, Y., & Patiniotakis, I. (2014, December). Underpinning a cloud brokerage service framework for quality assurance and optimization. In Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on (pp. 660-663). IEEE.

Verginadis, Y., Patiniotakis, I., Mentzas, G. (2013). D20.1 - State of the art and Research Baseline, Deliverable of the EU FP7 project Broker@Cloud.

Wang, H., Tianfield, H., & Mair, Q. (2014). Auction based resource allocation in cloud computing. Multiagent and Grid Systems, 10(1), 51-66.

Wang, S., Wang, Z., Li, C., Zhao, K., & Chen, H. (2016, September). Learn to Recommend Local Event Using Heterogeneous Social Networks. In Asia-Pacific Web Conference (pp. 169-182). Springer International Publishing.

Wang, W., Niu, D., Li, B., & Liang, B. (2013, July). Dynamic cloud resource reservation via cloud brokerage. In Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on (pp. 400-409). IEEE.

Weinhardt, C., Anandasivam, D. I. W. A., Blau, B., Borissov, D. I. N., Meinl, D. M. T., Michalk, D. I. W. W., & Stößer, J. (2009). Cloud computing–a classification, business models, and research directions. Business & Information Systems Engineering, 1(5), 391-399.

Weiss, A. (2007). Computing in the clouds. Computing, 16.

Wen, Y., Zhang, W., & Luo, H. (2012). Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones. Paper presented at the INFOCOM, 2012 Proceedings IEEE.

Windows Azure (2017) Microsoft. Retrieved from: https://azure.microsoft.com

Wolski, R., Gurun, S., Krintz, C., & Nurmi, D. (2008). Using bandwidth data to make computation offloading decisions. Paper presented at the Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on.

Wu, H., Huang, D., & Bouzefrane, S. (2013, October). Making offloading decisions resistant to network unavailability for mobile cloud collaboration. In Collaborative Computing:

Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on (pp. 168-177). IEEE.

Xiao, Y., Kalyanaraman, R. S., & Yla-Jaaski, A. (2008). Energy consumption of mobile youtube: Quantitative measurement and analysis. Paper presented at the Second International Conference on Next Generation Mobile Applications, Services, and Technologies.

Xing, T., Huang, D., Ata, S., & Medhi, D. (2012, October). MobiCloud: a geo-distributed mobile cloud computing platform. In Proceedings of the 8th International Conference on Network and Service Management (pp. 164-168). International Federation for Information Processing.

Xu, D., Song, G., Gao, P., Cao, R., Nie, X., & Xie, K. (2011, December). Transportation modes identification from mobile phone data using probabilistic models. In International Conference on Advanced Data Mining and Applications (pp. 359-371). Springer Berlin Heidelberg.

Ye, H., & Sugihara, G. (2016). Information leverage in interconnected ecosystems: Overcoming the curse of dimensionality. Science, 353(6302), 922-925.

Ye, Y., Xiao, L., Yen, I. L., & Bastani, F. (2011, July). Leveraging service clouds for power and qos management for mobile devices. In Cloud Computing (CLOUD), 2011 IEEE International Conference on (pp. 235-242). IEEE.

Yin, B., Shen, W., Cai, L. X., & Cheng, Y. (2015, June). A Mobile Cloud Computing Middleware for Low Latency Offloading of Big Data. In Proceedings of the 2015 Workshop on Mobile Big Data (pp. 31-35). ACM.

Yoshimura, Y., Sobolevsky, S., Ratti, C., Girardin, F., Carrascal, J. P., Blat, J., & Sinatra, R. (2014). An analysis of visitors' behavior in the Louvre Museum: A study using Bluetooth data. Environment and Planning B: Planning and Design, 41(6), 1113-1131.

ZABBIX (2001). Zabbix - enterprise-class open source monitoring solution. Retrieved from: www.zabbix.com

Zare Mehrjerdi, Y., & Nadizadeh, A. (2016). Using Greedy Clustering Method to Solve Capacitated Location-Routing Problem with Fuzzy Demands. International Journal of Industrial Engineering & Production Research, 27(1), 1-19.

Zenoss (n.d). Retrieved from: http://www.zenoss.org

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159-175.

Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R. P., Mao, Z. M., & Yang, L. (2010). Accurate online power estimation and automatic battery behavior based power model generation for smartphones. Paper presented at the Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis.

Zhang, P., & Yan, Z. (2011, September). A QoS-aware system for mobile cloud computing. In 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (pp. 518-522). IEEE.

Zhang, X., Kunjithapatham, A., Jeong, S., & Gibbs, S. (2011). Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. Mobile Networks and Applications, 16(3), 270-284.

Zhou, B., Dastjerdi, A. V., Calheiros, R. N., Srirama, S. N., & Buyya, R. (2015a). A context sensitive offloading scheme for mobile cloud computing service. Paper presented at the Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on.

Zhou, B., Dastjerdi, A.V., Calheiros, R., Srirama, S., & Buyya, R. (2015b). mCloud: A Context-aware Offloading Framework for Heterogeneous Mobile Cloud. IEEE.

Ziebart, B. D., Maas, A. L., Dey, A. K., & Bagnell, J. A. (2008, September). Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In Proceedings of the 10th international conference on Ubiquitous computing (pp. 322-331). ACM.

Ziegler, W., Jiang, M., & Konstanteli, K. (2011). Optimis sla framework and term languages for slas in cloud environment. OPTIMIS Project Deliverable D, 2.