A Monitoring System for Federated Clouds

Yahya Al-Hazmi*, Konrad Campowsky† and Thomas Magedanz*
*Chair of Next Generation Networks, Technical University Berlin, Berlin, Germany
*Email: {yahya.al-hazmi,thomas.magedanz}@tu-berlin.de
†Next Generation Network Infrastructure, Fraunhofer FOKUS Institute, Berlin, Germany
†Email: konrad.campowsky@fokus.fraunhofer.de

***Abstract*—Cloud computing mechanisms are steadily gaining significance in the field of IT infrastructure hosting and maintenance. To date, we have reached a point where a paradigm shift can be observed. An increasing number of IT players are rethinking the way the technical foundation of their offerings is operated from in-house datacenter hosting towards the outsourcing of infrastructure or services to specialized external cloud computing companies. With the emergence of more and more commercial cloud providers and offerings, combining resources from two or more providers - in order to benefit from factors like price differences, locality of resources, etc. - becomes increasingly feasible for consumers of cloud services. Therefore, it is desirable for such users to have an overarching monitoring system that aggregates a multitude of measurements from resources of different administrative domains in a unified manner. This paper introduces a comprehensive monitoring solution for federated clouds that provides data for both infrastructure providers and cloud users. This system does not only support monitoring resources from heterogeneous domains on both the network and infrastructure level, but moreover provides monitoring support that is able to operate across large numbers of end-to-end resources at the service as well as the application level. In this paper the design of this system as well as its implementation is discussed, and a validation of the system within the context of the European funded project BonFIRE is presented. The performance of the system is shown through the conduct of experimentation.**

***Index Terms*—Monitoring; Federated Clouds; Test Facilities; Federation;**

I. Introduction

The noticeable success of cloud computing encourages cloud service providers to create further enhancements to their facilities in order to offer services with a high level of customer satisfactions on the one hand and to adopt new technologies to build new business models on the other.

As a matter of fact, this emerging market spurs many cloud service players, which actually have different kind of resources with different level of quality of services, not only to compete but also to collaborate with each other. It is expected that such collaboration will have many advantages, such as complementation of the offered resources to improve resource utilization, merging multiple services in order to offer efficient end-to-end solutions required by the customers, and customers have the ability to create their own environments (e.g. Platform-as-a-Service) across multiple cloud domains. Moreover, it enables customers to combine resources and services from different cloud computing providers. This kind of collaboration across multiple cloud infrastructures that interoperate in a standardized manner is called resource federation.

Many cloud infrastructures are available from both business and research organizations who are providing cloud services (Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS)), and cloud-based test facilities for Future Internet research and experimentation respectively. Due to the high heterogeneity of such infrastructures in terms of resources, systems, tools, exchanged information, etc., the concept of standardization and federation is becoming of highly importance to avoid having isolated islands in the Future Internet. Note that when referring to a cloud user or a user in this paper, we are referring to a customer in commercial clouds or to an experimenter in cloud-based test facilities. Furthermore, when referring to a cloud environment, we are referring to a cloud service (e.g. PaaS, IaaS) in commercial clouds or to an experiment in test facilities.

All existing cloud computing services already run monitoring systems and will usually expose monitoring services to users as well in one form or another. However, cloud users could have resources from multiple cloud providers. Each of these clouds has its own monitoring system. Therefore it makes sense to have a comprehensive monitoring system that can operate seamlessly across cloud domain borders.

Monitoring is needed by cloud infrastructure providers and cloud users alike. Cloud providers need to monitor their physical resource like any other infrastructure owner does to ensure health and availability and Quality of Service (QoS) of their facilities. In addition to this, cloud providers have additional monitoring requirements due to the dynamic nature of their infrastructure, which enables them to perform sophisticated optimization of resource utilization. Cloud users on the other hand also benefit from comprehensive monitoring solutions since these are needed to facilitate features like service elasticity. In commercial clouds, customers need to monitor their resources (e.g. PaaS, IaaS) for performance evaluation, QoS and Service Level Agreement (SLA) validation, comparing the performance and the quality of resources from various cloud providers, etc. In test facilities, experimenters are interested in information about the actual utilization of their resources and also about the performance of their deployed technologies, protocols, or services.

The paper introduces a monitoring system for federated clouds targeting cloud providers and cloud-based test facilities.

The demand for having a common monitoring system for federated clouds has become a major interest, as disclosed recently by the Future Generation Computer Systems Journal (The International Journal of Grid Computing and eScience) through its Special Issue on Cloud Monitoring Systems, due to the fact that each application, platform or infrastructure has its own solution and this hinder the interoperability across federated clouds in terms of management, control charging, etc. Furthermore, many of the EU projects that started earlier 2012 are focusing on the interoperability, portability in the cloud domains, such as Compatibleone [1] and OpenCloudware [2]. However, providing monitoring information to cloud users about their resources deployed in different clouds is not considered.

The rest of the paper is organized as follow. In Section 2, the monitoring system is introduced. Section 3 presents the implementation and validation of the system. In Section 4, the system usability and performance are discussed through experimentation. Finally, the paper is concluded in Section 5.

II. System Design

This section presents the design of the monitoring system, its functionality, components and the interactions between them. The introduced monitoring system is targeting cloud providers and also cloud-based test facilities for Future Internet research and experimentation. It has a twofold objective. First, providing measurements and monitoring information for assuring the health and the performance of the federated cloud infrastructure and its internal services, and the stakeholder will then be cloud infrastructure providers. Second, providing service related measures that are consumed by cloud users to monitor their deployed services, and the stakeholders will therefore be cloud service (PaaS or IaaS) customers in commercial clouds and experimenters in test facilities.

The monitoring system is designed following a number of architectural principles:

- The monitoring support is based on the concept of "Monitoring-as-a-Service". The system is capable of providing monitoring on-demand. It leverages the paradigm of "everything-as-a-resource": if it's a resource, it can be instrumented (via specific resource adaptors called probes), and as such monitored. The monitoring data provided by the probes is aggregated in one aggregator. This aggregator offers an API, to be used by 3rd party software, and a graphical user interface (GUI) to display the monitoring data graphically.
- The introduced monitoring system has been designed in a way to allow user to choose whether or not to have Monitoring-as-a-Service. This decision can be taken while requesting cloud resources (PaaS or IaaS in commercial cloud or experiment in test facility).
- During the cloud environment deployment, the monitoring probes are deployed with the rest of the cloud resources, as they are just another deployable object. The deployment of the monitoring aggregator is also performed, which has the responsibility to aggregate all the measurements. All the probes are then automatically configured to report to the aggregator. The measured data can then be accessed via a simple web user interface, as well as an API. The system implementation outlined in this paper assumes that each cloud user has a single monitoring server running on one of the virtual machines (VMs) created as part of the cloud environment. This scheme on the one hand assures privacy of information and furthermore is guaranteed to work with a wide range cloud providers since it provides monitoring as an over the top service in a cross-domain manner.
- The user has the ability to have monitoring data permanently available after the expiration or deletion his/her cloud environment or only during its lifetime. Three cases to be discussed here:
  1) The user uses monitoring data to observe the correct progress of his/her cloud environment. There is no interest in keeping this information.
  2) The user decides to keep a trace of the cloud environment after its lifetime. As infinite disk space is not available, keeping all monitoring data indefinitely is not possible. Nonetheless, the full data for the last day (or whatever delay is compatible with the cloud providers' infrastructure) is available, for a limited time.
  3) The user knows that the complete monitoring data will be useful. In this case, while establishing the cloud environment, the user can ask for a persistent storage with a large enough storage size as needed.
- The user has two options on where to store monitoring data. It can be stored either inside the aggregator VM itself or on external storage resources. With the second option, the database of the aggregator is stored in an external (and permanent if required) storage that is mounted as an additional block device on the aggregator VM. This option enables more flexibility, the user can set, on-demand, the storage size for the monitoring data, and this data is also available after the cloud environments expiration or deletion. The default option is to create the aggregator with an external, permanent storage with a fixed size (e.g. 5GB).
- A group of user could work together on one cloud environment and have the same permissions, but they are using different credentials. In this case, the monitoring system is responsible to enable all group members to log in to the monitoring data using their own credentials. Furthermore, limited access to monitoring data with read-only permissions can be granted to other groups in the case of cooperation projects.

These architectural principles enable the monitoring system to provide flexible monitoring services. These services are supported at multiple levels:

- Infrastructure-level monitoring: it addresses the physical infrastructure of the federated clouds. Various metrics are addressed that are used for assuring the health and the

performance of the infrastructure, like: CPU-, memory-, disk-usage, number of VMs running on each physical machine, ingoing and outgoing traffic, etc.
- Cloud environment-level monitoring: monitoring metrics which address the cloud environment status at a certain moment of time and also the number of VMs per cloud environment.
- VM-level monitoring: status of individual computing resources, further metrics like: CPU, memory, storage, etc.
- Service-level monitoring: metrics which provide information about the state of the service, its performance and other service specific information.

As mentioned before, monitoring information is required by the cloud infrastructure providers for purposes of infrastructure Quality-of-Service and assurance the overall performance and the health of their infrastructures. The infrastructure monitoring is the infrastructure providers' responsibility. Through their native APIs, the infrastructures manage and monitor themselves using their own tools. To support the infrastructure monitoring, an infrastructure monitoring aggregator is required that gathers information regarding the whole infrastructure reported by the probes that are running in the all physical machines.

In addition to the VM-level and service-level monitoring, a user can also get partial information about the physical machines that host their VMs (infrastructure-level). Monitoring information about specific metrics is only provided, those of users' interest such as (CPU, memory, number of running VMs, network characteristics). Monitoring data of these metrics are fetched by user aggregators from infrastructure aggregators through APIs. However, when user VMs are deployed in multiple clouds, their infrastructure aggregators will then be accessed by the user aggregator to get the data. To realize the infrastructure monitoring service, the user aggregator has to be notified each time a VM is created, updated, or destroyed. One possible solution is the use of message queues that allow guaranteed notification delivery. The user aggregator acts as a client to subscribe to notifications sent when VMs' states are changed.

It is also possible for the user to associate monitoring results. In the user's view, the monitoring aspects are limited to the definition of probes, which then automatically deliver their measurement data to a graphical interface. This data is also accessible through the aggregator API.

The architecture of the monitoring system is shown in Figure 1. Monitoring probes, also called Agents (A) are deployed and configured on each of the user's VM resources. These agents are responsible for collecting monitoring metrics on their respective host. Their information is sent to a monitoring aggregator (User Aggregator), which holds all monitoring information, for direct consumption by the user (through an API or through a GUI). In addition, the User Aggregator fetches monitoring data about the physical infrastructure hosting the virtual components from the respective infrastructure's monitoring system (Infra Aggregator). A daemon is running

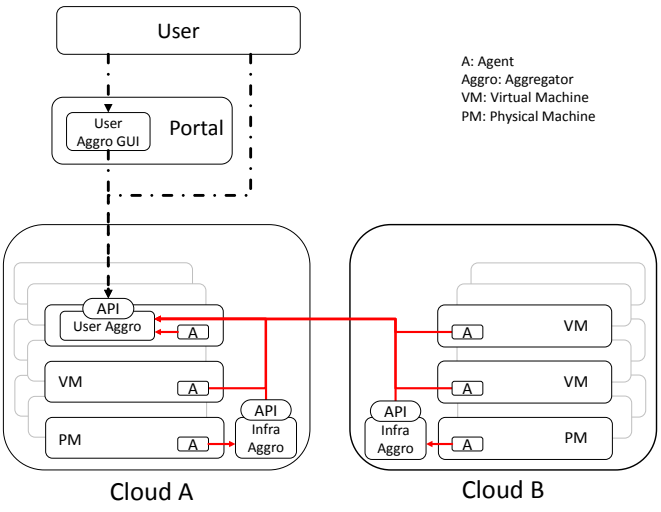

Fig. 1. Cloud Monitoring System

on the User Aggregator that is responsible for periodically fetching these metrics along with their timestamps and then storing the data into the database of the User Aggregator. This data is provided to the user (through the API or through the GUI of the User Aggregator) with its original timestamps.

The technical details of gathering metrics pose a scientific challenge themselves. Moreover, it is virtually impossible to predict all conceivable monitoring metrics that might be of interest for cloud users, especially when addressing the scientific users. Therefore, any comprehensive cloud monitoring solution must be extensible through user defined metrics.

III. Implementation and Validation

The monitoring system described in this work has been adopted by the EU BonFIRE Multi-Cloud Test Facility [3]. BonFIRE offers a multi-site cloud testbed that supports large scale testing of applications, services and systems over multiple, geographically distributed, heterogeneous cloud testbeds. At the core of BonFIRE are six geographically distributed testbeds (EPCC (UK), INRIA (France), HLRS (Germany), IBBT (Belgium), HP (UK) and PSNC (Poland)) that together offer around 350 computing cores with 700GB of RAM and 30TB of storage. An additional 3000 cores can be added to BonFIRE on-request. The involved testbeds are different from one another in terms of structure, networking features and resources. This heterogeneity is a key feature of the BonFIRE facility.

A. BonFIRE Monitoring System

The BonFIRE monitoring system provides users (experimenters) with access to performance metrics for the virtualized resources used in the experiment. It is also possible for experimenters to associate monitoring results with the experiment. The experimenter can define metrics to be monitored. Furthermore, experimenters can get partial monitoring information about the physical infrastructure as well.

Monitoring functionality 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 experiment VM images. To implement monitoring, BonFIRE has adopted the open source monitoring software Zabbix [4] that fulfills the requirements of the introduced monitoring system. This solution also supports alarms that trigger when predefined conditions are met, for example if the CPU load is over 90%. These alarms can be very useful as the triggers that control elasticity actions.

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 which 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.

The Zabbix aggregator collects monitoring information reported by the Zabbix agents in a database. It is the central repository where all information is stored and users and agents are managed. An aggregator can be automatically created for each experiment set up through the BonFIRE portal. Through the portal, experimenters have the ability to choose where to store the database, either inside the aggregator image or stored into an external storage resource that is attached to the aggregator as an additional, external disk. Experimenter's choice is passed then to its experiment aggregator (User Aggregator). This enables experimenters to store monitoring data in a flexible way, on-demand storage size, the possibility of reusing an external storage resource in further experimentations, etc.

In BonFIRE, an aggregator in each site (cloud infrastructure) is in charge of monitoring the whole physical infrastructure. However, experimenters can also get partial information about the physical machines that host their VMs. Experiment aggregators fetch monitoring data of predefined metrics relating to those physical machines from the infrastructure aggregators through their APIs. This is achieved through a permanently running daemon on the experiment aggregator. The notification about the changing of VMs' states is implemented in BonFIRE based on the message queue RabbitMQ [5]. A client running on the experiment aggregator is subscribed to it.

BonFIRE offers deployable packages with regard to monitoring which are called image packages. They represent the virtual machines images with the monitoring software already installed and configured. The following section describes the dependencies, installation and configuration steps for the aggregator and agent images packages.

B. BonFIRE Monitoring Images

The monitoring images provided by BonFIRE are Debian images. The following sections describe the steps necessary to deploy these images in the BonFIRE infrastructure using Open Cloud Computing Interface (OCCI) requests. As mentioned before, these images have the Zabbix software already installed and configured. Two images are available: monitoring aggregator image and monitoring agent image.

1) BonFIRE Aggregator Image: The Bonfire aggregator image contains both the monitoring aggregator and monitoring agent preinstalled. It is readily configured to support configuration of the aggregator through the contextualization information. This information is used for passing the requested monitoring services, such as infrastructure monitoring, having permanent storage, allowing the aggregator to monitor itself, etc. These are sent along with the aggregator creation request and will be used while booting the aggregator image. The BonFIRE Portal supports the automatic creation of an experiment's monitoring aggregator VM and also gives access to the experiment's monitoring aggregator's GUI or API. The BonFIRE Resource Manager [6] gives the experimenter access to the monitoring aggregator's API. However, the experimenter has the choice to create his experiment with monitoring support or not. If monitoring-as-a-service is desired, the experimenter should create an aggregator VM as part of the experiment. The choice where to locate it, in which site, is left to the experimenter.

a) Requirements: The aggregator image has a predefined id in the BonFIRE infrastructure. This id may change during the life time of the infrastructure. Therefore, before triggering the OCCI installation requests (more specifically creation of a compute resource) one must identify the image id from the BonFIRE Broker or BonFIRE Portal [6].

b) Installation: The following OCCI request when sent to the BonFIRE Broker triggers the creation of the image in the specified site.

```
<compute xmlns="http://api.bonfire-project.eu/doc/schemas/occi">
  <name>BonFIRE-monitor</name>
  <instance_type>small</instance_type>
  <disk>
    <storage href="/locations/site_name/storages/image_id"/>
    <type>OS</type>
    <target>hda</target>
  </disk>
  <disk>
    <storage href="/locations/site_name/storages/image_id"/>
    <type>disk</type>
    <target>hdb</target>
  </disk>
  <nic>
    <network href="/locations/site_name/networks/network_id"/>
  </nic>
  <context>
    <usage>monitoring_service_1;....;monitoring_service_N</usage>
  </context>
  <link href="/locations/site_name" rel="location"/>
</compute>
```

Using the context information delimited by <context> and </context> the aggregator is automatically configured based on the requested monitoring services.

2) BonFIRE Base Image: BonFIRE provides multiple VM images called BonFIRE base images. These images are used for booting virtual machines. Each base image contains an instance of the Zabbix agent software to serve as the monitoring Agent. It is configured to support configuration of the agent through contextualization information. BonFIRE's base images are preinstalled with monitoring agents preconfigured for some basic metrics such as CPU usage, memory, etc. Experimenters may configure other metrics when preparing their VM images. Additionally, the BonFIRE Resource Manager's OCCI API supports the specification of monitoring metrics in the contextualization section of a compute resource. The BonFIRE base images include software to read this contextualization information and configure the monitoring agents accordingly. Furthermore, the experimenter has also the ability to specify additional monitoring metrics to already running compute resources.

a) Requirements: As described for the aggregator image, the user needs to identify the allocated id to the monitoring agent image before the installation is performed.

b) Installation: The following OCCI request when sent to the BonFIRE Broker triggers the creation of the image in the specified site.

```
<compute xmlns='http://api.bonfire-project.eu/doc/schemas/occi'>
  <name>Monitoring Agent</name>
  <instance_type>small</instance_type>
  <disk>
    <storage href='/locations/site_name/storages/image_id'/>
    <type>DISK</type>
    <target>hda</target>
  </disk>
  <nic>
    <network href='/locations/site_name/networks/network_id'/>
  </nic>
  <context>
    <aggregator_ip>10.1.1.16</aggregator_ip>
    <metrics>
      <metric> metric_name, command, metric_attributes </metric>
      <metric> metric_name, command, metric_attributes </metric>
    </metrics>
  </context>
  <link href='/locations/site_name' rel='location'/>
</compute>
```

Using the context information delimited by <context> and </context> the Zabbix agent is automatically configured with the IP of the Zabbix server (monitoring aggregator) and the specific metrics defined by experimenter. These metrics must be configured on both the agent and server side.

IV. Experimentation

Various experiments taking advantage of the described monitoring system have been conducted on the BonFIRE facility or are currently in progress. These experiments are being conducted by industrial and academic researchers who are researching the behavior of cloud services under controlled conditions. Descriptions of selected experiments and their results in form of publications, tutorial, etc. are to be seen in [3]. In this paper, we will not discuss the details of specific measurements being taken, but rather shed light on its usability, and the extent of the benefit it brings to researchers.

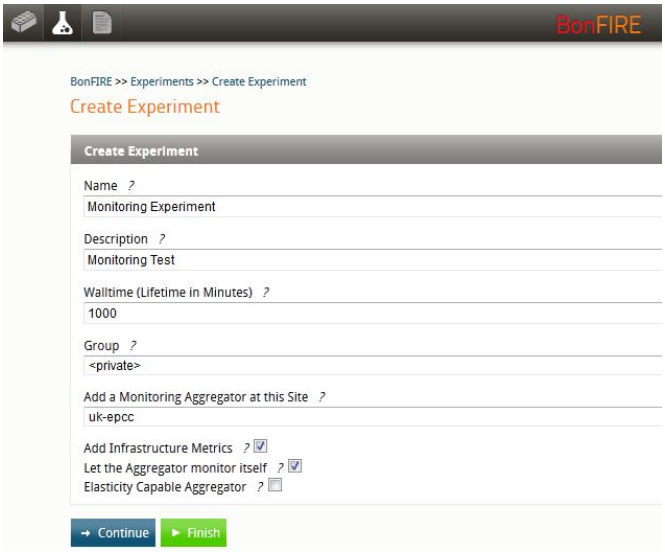

Fig. 2. Creating an Experiment with monitoring support

Many different metrics are measured by the system starting from low level resources (both physical and virtual) up to applications. Measurement data of these metrics and about partial of the infrastructure metrics that provide detailed information about the underlying physical infrastructures performance are provided to experimenters. For whatever purpose (e.g. QoS/SLA), some services (experiments) may need to monitor the CPU of the physical machines that host their VMs, how many VMs run on the same CPU, or what the load is on a specific CPU, etc. Through an experiment, we will show to what extent the performance and the benefit of the infrastructure monitoring is.

As described before, using monitoring services on BonFIRE implies having a computing resource (VM) that is running the BonFIRE aggregator image. Consequently, creating a BonFIRE experiment with monitoring support is facilitated by sending a number of individual OCCI requests to the BonFIRE resource manager API. These requests first create the actual experiment and subsequently deploy and configure a VM to serve as the monitoring aggregator. While this may seem as a complicated procedure at first, BonFIRE users may choose to perform it through the web-based BonFIRE portal, where monitoring support can be enabled for experiments merely by choosing which BonFIRE site the aggregator should be deployed to, as seen in Figure 2. Infrastructure monitoring support can be enabled or disabled on demand through the portal, as shown in Figure 2. BonFIRE offers its users elasticity-as-a-service that allow them to run elastic experiments, as seen in Figure 2. To support the elasticity service, an elasticity capable aggregator image is used that is supported with alarms that trigger when predefined conditions are met, for example if the CPU load is over a specific threshold. The configuration information needed to configure these alarms, also called triggers, is sent through the contextualization. In the experiment discussed here, we will not use elasticity service.

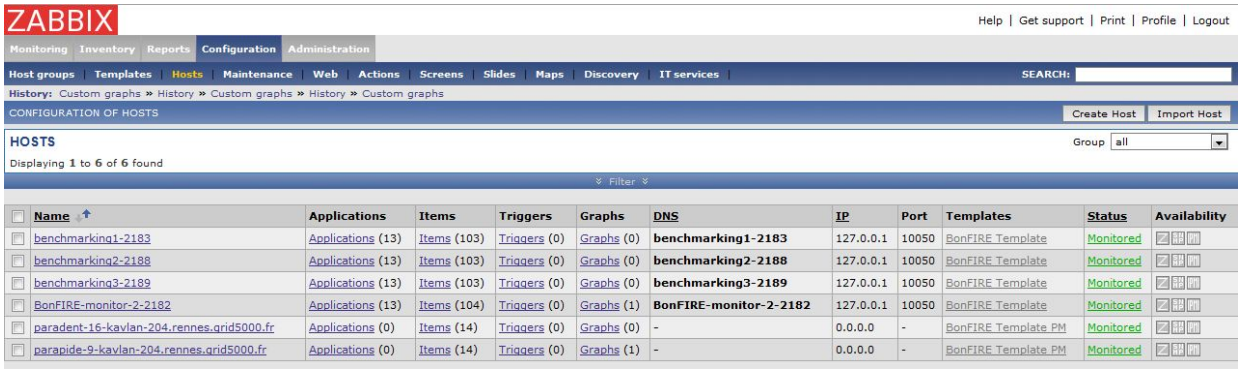

Fig. 3. Monitoring Aggregator User Interface

At this point, an experiment with monitoring but without elasticity support is created. As mentioned before, the BonFIRE monitoring aggregator is based on the Zabbix monitoring framework. The aggregators web-based user interface can be accessed seamlessly through the BonFIRE portal. This user interface is shown in Figure 3. The experimenter can then start creating the required experiment VMs that will be automatically monitored along with their physical machines. Their data will be viewed through the aggregator GUI as well.

BonFIRE offers reservation of physical machines called clusters for exclusive use. In this experiment, a cluster is reserved to ensure that only experimenter VMs are running on it. On this cluster, three VMs are created. At this point, the experiment comprises one aggregator VM running on a physical machine (anywhere) and three VMs running on the reserved cluster as seen in Figure 3. On the three VMs, benchmarking is run not for testing performance but only to investigate and show the behavior of performance metrics like CPU load within the VMs and the cluster. The IOzone benchmarking [7] is used for this purpose although it is practically used for testing file I/O performance on various filesystems. Nevertheless, the operations performed by the benchmark consume processing power and the CPU load will be varying based on the number of operations being executed by the benchmarking and the filesystem that is employed. The benchmarking was run for more than two days, Figure 4 shows the behavior of the CPU load within the physical cluster represented by the blue line, and the average of the CPU loads within the three virtual machines represented by the black line. The result shown in the first graph is for one hour while in the second graph is for twelve hours. As expected, loads are almost similar which indicates that experimenters can get meaningful real-time information about the underlying infrastructure that can be used as support for suitable decision-making. Furthermore, from the results, it is clear that the CPU processing consumed by monitoring probes (Zabbix agents) is almost negligible.

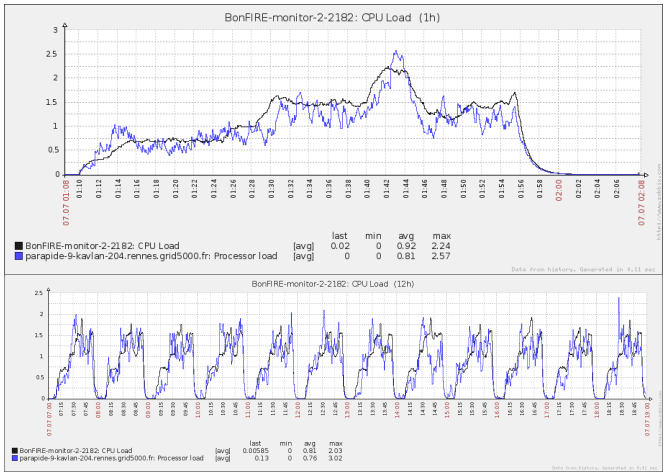

Fig. 4. CPU Load within a physical machine (blue) and the average CPU Load within three virtual machines running on the physical machine (black)

V. Related Work

Monitoring is a fundamental part of any management system. In the domain of cloud computing, examples for cloud monitoring systems or systems that are used within such are EVEREST [8], Ganglia [9], Nagios [10], Groundwork [11], MonALISA [12], and Zabbix [4]. Furthermore, there are several monitoring architectures already deployed in cloud environments [13] [14] [15] [16] [17] [18] [19]. Monitoring in the OPTIMIS project [13] is one of the OPTIMIS Toolkit's software components that is used to provide monitoring information about the virtual and the physical resources of the involved cloud infrastructures and services in order to support self-management and optimization processes. The service oriented monitoring framework in [14] is using the monitoring Toolkit Nagios, which is extended through the implementation of NEB2REST to interact with a RESTful web service for monitoring resources and services. The monitoring frameworks in [15] (which is based on the open source Globus Toolkit 4) and in [16] are used for QoS measurements at application and infrastructure levels for supporting real-time QoS guarantees. The authors in [17] introduced a private cloud monitoring system (PCMONS), which is based on the open source Nagios tool. The elastic monitoring framework introduced in [20] enables monitoring resources from low-level metrics from operating systems to higher level application-

specific metrics derived from services.

However, these systems and architectures are addressing the monitoring of cloud environments but not federated clouds in which a large number of resources from heterogeneous infrastructures are offered to customers.

The cloud related project RESERVOIR has introduced a monitoring system fit to its needs as well [21]. Its main functionality is to provide wide monitoring information about services deployed in federated clouds for service management purposes, such as service billing, service elasticity, access control, SLA management, etc. However, this system does not consider providing monitoring information to cloud customers. The Amazon monitoring system CloudWatch [22] on the other hand provides monitoring data to customers about their running services, rather than providing data for infrastructure and service management.

In contrast to the above mentioned systems, we introduce a comprehensive monitoring solution for federated clouds that provides data for both infrastructure management and cloud customers as well. The introduced monitoring system does not only support monitoring on network and infrastructure levels from heterogeneous domains, but also provide monitoring support that run across large populations of end-to-end resources at the service and application levels.

VI. Conclusion and Future Work

This paper introduces a comprehensive monitoring solution for federated clouds that provides data for both infrastructure providers and cloud users. This system supports monitoring resources (both physical and virtual) from heterogeneous domains on both the network and infrastructure level, and moreover provides monitoring support at the application level. In this paper the design of this system as well as its implementation is discussed, and a validation of the system within the context of the European funded project BonFIRE is presented. The usability and the benefit of the system are discussed in form of conducting experimentation on BonFIRE facility. For future work, this system will be the first step toward developing a generic monitoring system for federated Future Internet infrastructures not only in cloud domains but also others. This generic system will be developed within the context of the upcoming European funded project Mobile Cloud Networking. Deciding how to collect measurements to minimize the impact on the experiments themselves, and how to analyze these data to arrive at scientifically sound conclusions remain a large challenge which is left for future work as well.

Acknowledgment

This work was undertaken in the context of the BonFIRE project which is funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257386. The authors would like to thank all BonFIRE development team for their contribution and support to deploy the introduced monitoring system, in particular, Irina Boldea, Abdulrahman Hamood, and Maxence Dunnewind.

References

[1] CompatibleOne, "Compatibleone project," Website, available online at www.opencloudware.org, last visited on July 6, 2012.

[2] OpenCloudware, "Opencloudware project," Website, available online at www.compatibleone.org, last visited on July 6, 2012.

[3] BONFIRE, "European funded project bonfire - testbeds for internet of services experimentation," Website, available online at www.bonfire-project.com, last visited on July 6, 2012..

[4] ZABBIX, "Zabbix - enterprise-class open source monitoring solution," Website, available online at www.zabbix.com, last visited on July 6, 2012.

[5] RabbitMQ, "Rabbitmq," Website, available online at www.rabbitmq.com, last visited on July 6, 2012.

[6] A. C. Hume *et al.*, "Bonfire: A multi-cloud test facility for internet of services experimentation," in *Testbeds and Research Infrastructures for the Development of Networks & Communities, TRIDENTCOM 2012. 8th International Conference on*. IEEE, 2012, pp. 1–13.

[7] I. Benchmarking, "Iozone benchmarking," Website, available online at www.iozone.org, last visited on July 6, 2012.

[8] G. Spanoudakis, C. Kloukinas, and K. Mahbub, "The serenity runtime monitoring framework," *Security and Dependability for Ambient Intelligence*, pp. 213–237, 2009.

[9] Ganglia, "Ganglia monitoring system," Website, available online at www.ganglia.sourceforge.net, last visited on July 6, 2012.

[10] NAGIOS, "Nagios monitoring tool," Website, available online at www.nagios.org, last visited on July 6, 2012.

[11] GroundWork, "Groundwork," Website, available online at www.gwos.com, last visited on July 6, 2012.

[12] MonALISA, "Monalisa: Monitoring agents uasing a large integrated services architecture," Website, available online at monalisa.caltech.edu/monalisa.htm, last visited on July 6, 2012.

[13] J. Tordsson *et al.*, "Towards holistic cloud management," in *European Research Activities in Cloud Computing*, D. Petcu and J. L. Vazquez-Poletti, Eds. Cambridge Scholars Publishing, 2012, pp. 122–150.

[14] G. Katsaros, R. Kübert, and G. Gallizo, "Building a service-oriented monitoring framework with rest and nagios," in *Services Computing (SCC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 426–431.

[15] G. Katsaros, G. Kousiouris, S. Gogouvitis, D. Kyriazis, and T. Varvarigou, "A service oriented monitoring framework for soft real-time applications," in *Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–4.

[16] G. Katsaros, R. Kübert, G. Gallizo, and T. Wang, "Monitoring: A fundamental process to provide qos guarantees in cloud-based platforms," *Cloud computing: methodology, systems, and application*, pp. 327–339, 2011.

[17] S. De Chaves, R. Uriarte, and C. Westphall, "Toward an architecture for monitoring private clouds," *Communications Magazine, IEEE*, vol. 49, no. 12, pp. 130–137, 2011.

[18] G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. Fitó, and D. Henriksson, "A multi-level architecture for collecting and managing monitoring information in cloud environments," in *Cloud Computing and Services Science, 2011 CLOSER. 1st International Conference on*. IEEE, 2011, pp. 1–4.

[19] G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. Fitó, and D. Espling, "An integrated monitoring infrastructure for cloud environments," *Cloud Computing and Services Science*, pp. 149–164, 2012.

[20] B. Koenig, J. M. A. Calero, and J. Kirschnick, "Elastic monitoring framework for cloud infrastructures," *IET Communications*, to appear in the IET Digital Library.

[21] S. Clayman, G. Toffetti, A. Galis, and C. Chapman, "Monitoring services in a federated cloud - the reservoir experience," in *Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice*, M. Villari, I. Brandic, and F. Tusa, Eds. IGI Global, 2012.

[22] CloudWatch, "Amazon cloudwatch," Website, available online at www.aws.amazon.com/cloudwatch, last visited on July 6, 2012.