2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery

QoS Management in Cloud@Home Infrastructures

Salvatore Distefano, Antonio Puliafito
University of Messina, Messina, Italy
Email: sdistefano, apuliafito@unime.it

Massimiliano Rak, Salvatore Venticinque
Second University of Naples, Aversa, Italy
Email: massimiliano.rak, salvatore.venticinque@unina2.it

Umberto Villano, Antonio Cuomo
University of Sannio, Benevento, Italy
Email: villano, antonio.cuomo@unisannio.it

Giuseppe Di Modica, Orazio Tomarchio
University of Catania, Catania, Italy
Email: gdimodica, tomarchio@diit.unict.it

***Abstract*—Cloud is strongly emerging as the new deal of distributed computing. One of the reason behind the Cloud success is its business/commercial-oriented nature, proof of its effectiveness and applicability to real problems. There are actually a lot of open-private Cloud infrastructures aiming at providing dynamic-on demand resource provisioning according to the IAAS paradigm. To this purpose they usually apply a best effort policy, without taking into account service level agreements (SLA) and related quality of service (QoS) requirements. In such a context, the main goal of Cloud@Home is to implement a volunteer-Cloud paradigm which allows to aggregate Cloud infrastructure providers. In this work we specifically focus on SLA-QoS aspects, describing how to provide SLA based QoS guarantees through Cloud@Home on top of non-QoS oriented Cloud Providers. Aim of the paper is to demonstrate how Cloud@Home can fulfil such goal, providing and specifying the architecture, the algorithms and the components that implement SLA-QoS management features.**

***Keywords*-Cloud; Volunteer Computing; QoS; IAAS.**

I. Introduction

In the last few years the cloud computing paradigm has emerged as one of the most interesting innovations in the ICT field. Cloud computing is made up of rather well-known and established technologies and solutions, which can be combined in disparate ways so as to offer solutions with different flavors. Twelve definitions of “cloud computing” are reported in [1], but the most cited is that by NIST [2]. In fact, the cloud concept hinges on the delegation to the network not only of resource provisioning, but also of all forms of infrastructure management. Its great success is due to both the maturity of underlying technologies (virtualization, Web 2.0, SOA model) and to the desire of organizations and of single users to give up the management of complex and critical hardware-software infrastructures, adopting a convenient and troubleless pay-per-use paradigm for all their computing and storage needs.

In this paper, the stress will be on the Infrastructure as a Service model (IaaS), in that clouds are used for providing infrastructure components as computational units, storage and communication systems. An IaaS *Cloud Provider* (CP) heavily relies on a virtualization system, since the leased computing resources are single or multiple Virtual Machines (VMs). The user has the possibility to choose between a variety of pre-built virtual images with different OS and pre-installed software, or even to supply his own image. Furthermore, he has also full administration rights on the VM(s), thus making it possible a fine tuning of configurations and software.

At the state of the art, there exist multiple frameworks able to offer IaaS services, such as Eucalyptus [3], OpenNebula [4], Nimbus [5], PerfCloud [6], Clever [7]. The common denominator of these, as well of other proprietary solutions (i.e., Amazon EC2, Microsoft Azure, ...) is the exploitation of powerful and reliable underlying computing infrastructures. These typically consist of a single or multiple datacenters, whose computing and storage resources are virtualized and leased to the cloud users. A completely different approach is instead used in *Cloud@Home*, a project funded by Italian Ministry for Education and Research [8]. Cloud@Home (briefly, *C@H*) aims at building an IaaS Cloud Provider using computing and storage resources acquired from volunteer contributors. These resources are likely to be unused PCs, small clusters of workstations from research laboratories and, more in general, commercial-off-the-shelf systems (COTS). Hence, the basic infrastructure is made up of intermittently-available computing resources over fairly unstable networks, and so its use for implementing a cloud IaaS provider (the C@H provider) is a particularly challenging problem. The objectives of the Cloud@Home project and its rationale, along with the pros and cons, the limits and the application contexts of its proposed architecture are dealt with in [9].

One of the strong points of the Cloud@Home proposal is the provision of a system for managing the quality of the leased services, system which is thoroughly described in this paper. As mentioned before, C@H providers aim at delivering QoS on top of a physical infrastructure made up of volunteered resources. These are not suitable to guarantee any QoS requirement, starting from the most basic of all quality indexes, availability. In fact, volunteered resources

978-0-7695-4557-8/11 $26.00 © 2011 IEEE
DOI 10.1109/CyberC.2011.40

are by their nature transient, not to mention their possible unavailability due to network faults. Stated another way, in a highly dynamical and volunteered environment such as C@H, just keeping the leased resources "alive" is not a trivial problem. In the C@H framework, every time that a user requests a service, the QoS to be guaranteed is negotiated between the user and the C@H Provider, and it is formally agreed upon through a Service Level Agreement (SLA). In this step, the C@H Provider makes use of tools and internal services to evaluate and to predict the sustainability of the QoS requested, as well as to identify the best combination of resources to be leased to the user. Moreover, the system is constantly monitored, triggering suitable recovery procedures to guarantee the agreed QoS level in the presence of performance faults.

In this paper the focus will be on the negotiation and on the enforcement of SLA in C@H. The issues dealt with include performance fault detection and recovery. The analysis and the solutions that will be presented here are valid more in general for any hierarchical scenario where an upper-level cloud provider wishes to guarantee QoS requirements even if it relies on lower-level cloud providers that do not offer QoS management.

The paper is organized as follows. The next section briefly sketches the Cloud@Home architecture. Then, in Section III, the components devoted to the management and enforcement of QoS are presented and discussed. Section IV presents use cases related to the start-up and the SLA/QoS management of the infrastructure. Related work is summarized in Section V, which is followed by our conclusions.

II. The C@H architecture

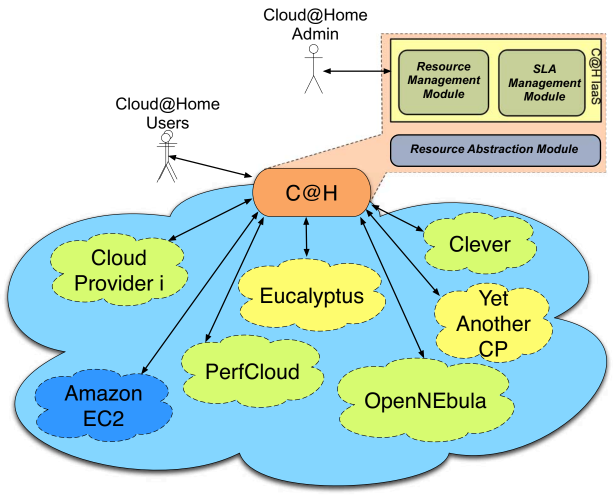

Figure 1: The C@H system architecture.

Cloud@Home provides a set of tools enabling the composition of a new, enhanced provider of resources (namely, a *C@H Provider*) which is not yet another classic Cloud Provider, but rather puts itself in acting as a resource aggregator (see Figure 1). It collects resources from several Cloud Providers (either public or private) merging different technologies and heterogeneous resource management policies, and offers such resources to the users in a uniform way. According to such viewpoint three actors can be identified:

- *Cloud Provider:* offers resources (basically, hardware systems such as computing and storage elements) according to the IaaS cloud service model. Any private or public entity can provide its own resources to the C@H Provider.
- *C@H Admin:* builds up and manages the C@H Provider. The C@H Admin is the manager of the C@H infrastructure and, in particular, is responsible for its activation, configuration and maintenance. There can be just one C@H Admin per C@H Provider.
- *C@H User:* interacts with the *C@H Provider* in order to obtain the resources.

The focus of this paper is on the role of the C@H Provider as aggregator of resources that are voluntarily provided by their owners. Volunteer resources (i.e. resources that can join or leave the Cloud) bring into the infrastructure a certain degree of unpredictability which may cause a degradation of the QoS at provision time. The C@H Provider's goal is just to keep this service level as high as requested by the C@H User, by enforcing countermeasures to face the intrinsic uncertainty which is typical of such kind of environments. To this end the C@H Provider offers, along with basic services for the collection of volunteer-based resources, a set of tools and services for the negotiation, monitoring and enforcement of the QoS.

The overall set of functionalities supplied by the C@H Provider is organized in three modules, that are depicted in the top-right part of the Figure 1. The **Resource Abstraction Module** implements and provides tools and mechanisms to 1) interface to IaaS Cloud Providers willing to offer resources and 2) interconnect them into a unique C@H Provider. The task of this module is to hide the heterogeneity of resources collected from the specific Cloud Providers (computing/storage elements) and to offer the C@H User a uniform way to access them. The **Resource Management Module** is entrusted with the management (discovery, enrollment, activation, VM migration, etc.) of the resources. The **SLA Management Module** is in charge of negotiating with the C@H User the quality level of the requested resources, then formalized through a Service Level Agreement.

Each module is composed of *C@H Components* implementing specific functionalities. C@H Components are able to interact with each other through standardized, service-oriented interfaces. Such a choice enables flexible component deploying schemes. A single C@H Component can even be offered as a customized virtual machine hosted by any Cloud Provider. In this way the presented approach

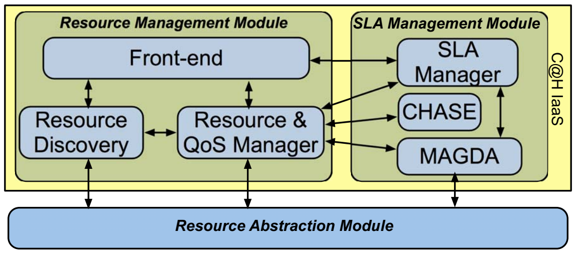

Figure 2: The C@H QoS/SLA management system architecture.

can be considered as a *Platform-as-a-Service* Cloud system which offers basic building blocks that can be easily composed to create a customized C@H Provider supplying resources in an *Infrastructure-as-a-Service* fashion.

III. QoS MANAGEMENT

The aim of C@H is to provide tools and mechanisms for the management of the QoS to be delivered to the C@H Users. As introduced above, the focus in this paper is on a specific QoS parameter, *resource availability*. However, the description of the algorithms and mechanisms implemented in C@H to manage QoS requirements and related SLA policies is more general, and can be applied to other QoS parameters.

The enforcement of the SLA/QoS encompasses the following activities:

- **negotiation -** C@H Users can negotiate the QoS levels to be provided and, upon termination of the process, receive a formal guarantee (in the form of an SLA) that the agreed QoS levels will actually be delivered;
- **monitoring -** the non-functional parameters contributing to the overall QoS are constantly monitored to ensure that none of the SLA terms is being violated;
- **recovery -** recovery actions are triggered whenever the QoS is about to be violated and countermeasures are to be taken in order to bring the QoS back to "safe" levels;
- **termination -** termination simply frees the resources leased to the user.

The process starts with the negotiation of the QoS between the parties (C@H User and Provider). In particular, the C@H User submits an SLA proposal to the C@H Provider. If the C@H Provider rejects the proposal, the process terminates. Otherwise, the SLA is formally signed by the parties. The QoS constraints are extracted from the SLA document and are constantly monitored. Had an SLA violation to be detected, the recovery task would be triggered. If no recovery is possible, the service provision terminates; otherwise, the QoS is reset to its initial level and monitored against any possible new faults.

From an architectural point of view, the C@H module decomposition shown in Figure 2 takes inspiration from the SLA@SOI's SLA reference model [10]. More detailed, the SLA Management Module is made up of three components: the *SLA Manager*, the *Cloud@Home Autonomic Service Engine* (CHASE) [11] and the *Mobile Agent Based Grid Architecture* (MAGDA) [12]. The SLA Manager is the core component, i.e., the orchestrator of the entire QoS management process, supported by the Resource and QoS Manager (described later). They are both in charge of supporting the negotiation phase and of coordinating the monitoring and the recovery phases. Moreover, the SLA Manager aggregates and advertises the templates of the SLAs published by the Cloud Providers interacting with C@H. SLA templates can then be searched for by C@H Users and filled according to specific non-functional requirements (e.g., resource availability level). Once the C@H User has submitted his requirements, the negotiation process can start.

The Resource Management Module is also composed of three functional blocks: the *Resource Discovery*, the *Frontend* and the *Resource and QoS Manager*. The Resource Discovery collects and indexes the resources. The Frontend gathers the C@H Users' requests and dispatches them to the appropriate module. The Resource and QoS Manager (RQM) manages and coordinates the C@H components, also tracking the state of the system and of the requests. More details on this topic can be found in [9].

While user-provider interactions are governed by the WS-Agreement protocol [13] rules implemented inside the SLA Manager, the negotiation process is supported by CHASE by means of a simulation-based performance prediction tool that assesses the sustainability of C@H Users' requests. CHASE implements the SLA@SOI Service Evaluation component, able to consider both the application and target resources' information in order to identify the best set of resources to be acquired and the best way to use them from the application point of view [11].

Based on the CHASE prediction, the SLA Manager decides either to accept the C@H User's proposals or to reject it. Upon rejection, the C@H User can issue a new proposal, in the form of a counter offer with less demanding QoS parameters that the system could likely sustain. At the end of a successful negotiation, an SLA is signed by both the parties (C@H User and C@H Provider). The process that leads to the sign of the agreement strictly follows the WS-Agreement protocol and therefore both the SLA templates and the SLA documents thus obtained are fully compliant to the WS-Agreement format specification.

After the negotiation, the control is handed over to the *RQM* component. In this phase the RQM, acting as a Service Manager, triggers the actual service provision, i.e., takes care of searching and of activating the resources that will have to support the C@H User's functional and non-functional requirements, as expressed in the SLA. After that, the RQM invokes a service to set-up an *ad hoc* infrastructure dedicated to the monitoring of the resources.

The *MAGDA* component (corresponding to the SLA@SOI's Manageability Agent and Monitoring System) carries out the monitoring of the leased resources. MAGDA provides a service able to perform different kinds of monitoring tasks, from simple local data sampling (current CPU usage, memory available, . . .) to distributed monitoring implemented through a mobile agent platform. Moreover, the mobile agents are able to update an archive of the collected measurements, in order to perform historical data analysis. It is important to point out that MAGDA works on top of the delivered VMs, consuming a very small fraction of their memory and computational power (see [12] for details).

Whenever the monitoring infrastructure detects a possible imminent violation of the SLA, the RQM is alerted and solicited to react to the QoS degradation (similarly to the SLA@SOI's Provisioning and Adjustment Components). The RQM elaborates the appropriate countermeasures (re-configure, re-allocate, migrate the VM instance) and enforces them to recover from the potentially "unsafe" conditions. In this context, RQM may invoke again CHASE in order to collect new predictions for the selection of new available resources in order to migrate the VM, if necessary.

Finally, when the service completes according to the SLA, or the provision has to be forcedly interrupted because of the impossibility of recovering from a fault, the termination phase is invoked by the RQM. In this phase, all the resources are released and their status updated.

IV. Use Case Scenarios

This section describes in details, through a practical/use-case based approach, how to set up a C@H Provider and how this latter negotiates and enforces QoS on top of resources voluntarily shared by their Cloud Providers. In subsection IV-A, we first show how a C@H Admin can build and set up the C@H components that implement the C@H infrastructure according to the IaaS service he wants to provide (with or without fault tolerance, QoS-SLA management, . . .). In subsection IV-B and subsection IV-C the QoS negotiation and enforcement dynamics are described, respectively. As regards the described scenarios, the target resources taken into account are academic clusters hosting a cloud middleware (e.g., Eucalyptus [3], Nimbus [5], Perfcloud [6], Clever [7]) able to provide Virtual Clusters (VCs). We also assume that such clusters have a reliable front-end, while compute nodes are unreliable since, for example, they might crash due to power outages or they might be periodically reconfigured/updated and become temporarily unavailable.

The scenario we consider implements a C@H infrastructure able to satisfy C@H User requests of VCs including specific QoS requirements. In particular, as introduced before, the QoS is expressed in terms of *availability* i.e., the fraction of time (percentage) the node must be up and reachable with respect to a 24 hours base time. In order

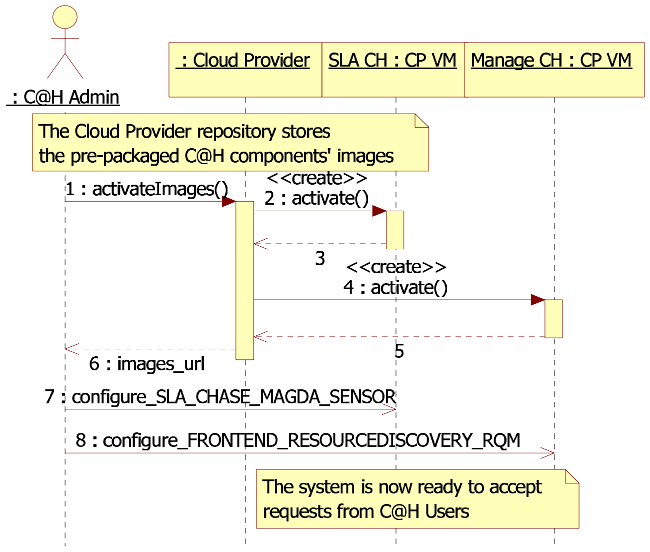
: C@H Admin
: Cloud Provider
SLA CH : CP VM
Manage CH : CP VM
The Cloud Provider repository stores the pre-packaged C@H components' images
1 : activateImages()
<<create>>
2 : activate()
3
<<create>>
4 : activate()
5
6 : images_url
7 : configure_SLA_CHASE_MAGDA_SENSOR
8 : configure_FRONTEND_RESOURCEDISCOVERY_RQM
The system is now ready to accept requests from C@H Users

Figure 3: Cloud@Home system setup

to do that we have to specify that all the nodes belong to the same Cloud Provider. Actually C@H is able to pick resources from different providers, but in this specific case all resources are acquired by the same provider. Finally, the VC images host the MAGDA agent-based monitoring service.

A. System Setup

The C@H Admin is the actor responsible for the deployment and the setup of the C@H infrastructure. As already pointed out in section II, the C@H components are provided with service-oriented interfaces. Such a choice enables flexible deployment schemes. Inspired by the "as-a-service" paradigm, individual (or group of) components can be themselves packaged together and run within virtual machines offered by any (even commercial) CP.

The C@H Admin has therefore to select the components of the C@H IaaS he wants to provide. In order to implement the considered scenario, it is necessary to build up a C@H infrastructure able to deliver SLA/QoS services. Thus, it is necessary to build up a C@H IaaS with all the modules and components shown in Figure 2.The C@H project's repository stores ready-to-run VMs (in the formats suitable for the different Cloud Providers) pre-packaged with the C@H components responsible for QoS management. Before deploying the system, such images have to be retrieved from the repository. In the specific scenario taken into account, the C@H Admin needs the following images:

- **SLA_CH:** packaged with the *SLA Manager*, *CHASE* and *MAGDA* components.
- **Manage_CH** packaged with the *Frontend*, *Resource Discovery* and *RQM* components.

- **VCFE** and **VCnode**, i.e., the images of the virtual cluster frontend (VCFE) and virtual cluster node (VCnode) respectively.

Only one instance for each of the first two images (**SLA_CH** and **Manage_CH**) has to be deployed. The VCFE and VCnode images are instead used in the next scenario.

The sequence diagram of Figure 3 shows the setup workflow. The C@H Admin invokes the *activateImages* primitive on the target Cloud Provider (step 1) that retrieves and launches the **SLA_CH** image (step 2). Afterwards, the Cloud Provider retrieves and runs the **Manage_CH** (step 4). The URLs of the just created components are forwarded back to the Admin through step 6.In steps 7 and 8 the C@H software components running on the just bootstrapped images are configured. The C@H system is now up and ready to accept requests from C@H Users. It is important to remark that the system setup process does not depend on the Cloud Provider chosen to deploy the C@H components.

B. Negotiation

Let us assume that the VC images are already available on the target Cloud Provider. In Figure 4 the (one-shot) negotiation and the cluster activation processes are depicted. The C@H User initially requests the C@H's SLA templates (step 1–4). WS-Agreement compliant templates are employed to guide the C@H User in the specification of the desired quality level. In this use case the filled template will look like the following:

Listing 1: Resource and Availability request in WS-Agreement

```
    <ws:ServiceDescriptionTerm
      ws:Name="CLUSTER REQUEST"
      ws:ServiceName="SET VARIABLE">
      <mod:Cluster xmlns:
         mod="http://occi-wg.org/model"
       <Compute>
         <architecture>x86</architecture>
         <cpuCores>4</cpuCores>
         [...]
         <title>compute1</title>
      <Compute>
      <Compute>
         <architecture>x86</architecture>
       [...]
         <title>compute2</title>
      <Compute>
     [...]
     </Cluster>
    </ws:ServiceDescriptionTerm>
   [...]
    <wsag:GuaranteeTerm
      wsag:Name="Availability"
      wsag:ServiceScope="C@H Cluster">
      <wsag:Variables>
        <wsag:Variable
          wsag:Name="NodeAvailability"
        wsag:Metric="ch:availability">
      </wsag:Variable>
      <wsag:ServiceLevelObjective>
        97
      </wsag:ServiceLevelObjective>
      </wasg:Variables>
      [..]
    </wsag:GuaranteeTerm>
```

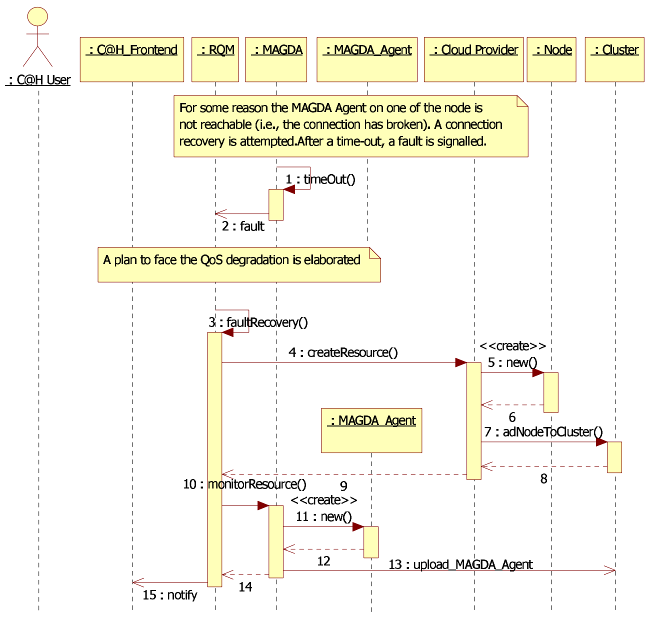

Figure 5: Enforcing the QoS in C@H

The section ServiceDescriptionTerms specifies the functional parameters. The reader may notice that the request complies with the OCCI specification format. In the GuaranteedDescriptionTerms section the C@H User specifies the required QoS, which in this specific case is the node availability (97 %).

The one-shot negotiation is performed in steps 5–10. The RQM, which coordinates the negotiation process, relies on the CHASE service to assess the sustainability of the SLA proposal. CHASE is aware of the cluster topology and maintains information on technical features such as the *mean time to failure* (mttf), the *mean time to repair* (mttr), i.e., the time to recovery from a node fault, as well as the time for booting the cluster up(t_s). Upon a given request, CHASE performs a simulation assessing if the request is compatible with the considered mttf and the mttr. In the positive case an acknowledgement is sent back to the RQM (step 9) which is in turn forwarded to the SLA manager (step 10). Otherwise, the maximum service level that the system is able to provide according to the simulation results is returned.

Upon positive acknowledgement (step 11), the RQM searches for the CP offering resources (cluster of x nodes) that are able to satisfy the C@H User's functional request (steps 12 and 13). Once the provider is identified the RQM activates the cluster (steps 14–17), whose URL is provided back to the C@H User through the Frontend (step 18). An agent-based monitoring infrastructure, instructed with the QoS target to be accomplished, is also set up (steps 19–

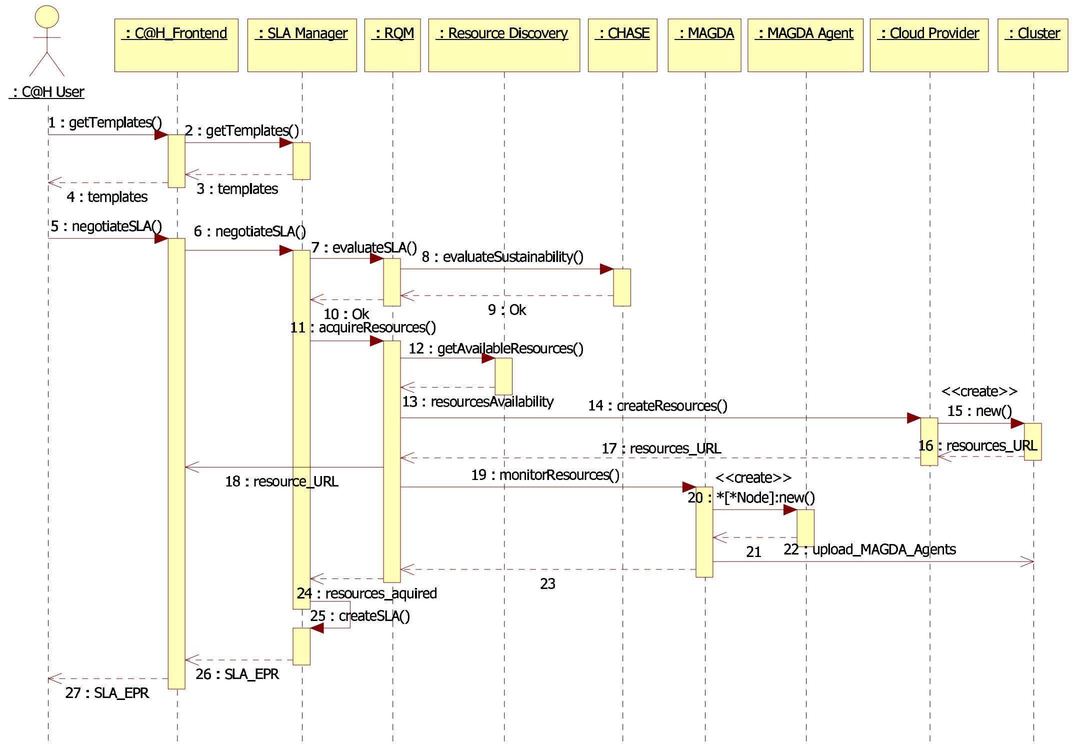

Figure 4: Resource negotiation and setting in C@H

24). Finally, the SLA is created (step 25) and its endpoint reference (EPR) is notified to the C@H User (steps 26-27).

C. Fault recovery and SLA enforcement

Let us suppose that the virtual cluster of nodes is up, and that the MAGDA agents are running on the virtual nodes and reporting the observed status to the C@H MAGDA component. The use case considered here, described in Figure 5, concerns a scenario in which a node/VM of the VC suddenly becomes unreachable.

In real situations this could be due to hardware/software outages, networking issues or even machine reconfiguration or reboot processes.

Since the C@H MAGDA component does not receive status reports from the MAGDA agent running on the unavailable node, a fault is believed to have occurred, and a recovery procedure is triggered on the cluster frontend in order to ascertain what is actually going on.

If the MAGDA agent on the faulty node does not reply within a given timeout (whose length is tailored on the SLA's requested availability parameter) the C@H MAGDA sends a *critical recovery* message to the RQM (step 2). According to the fault alert and to the QoS requirements established by the SLA policy to be enforced, the RQM decides if and when to begin the fault recovery process (step 3). The internal recovery procedure activates a new node/VM on the Cloud Provider and integrates it into the existing cluster (steps 4–9). Finally, a new MAGDA agent is sent and launched into the new node (steps 10–14), and the successful QoS adjustment is notified to the C@H Frontend (step 15).

D. C@H Testbed

In order to implement the scenarios described above we have set up a real testbed made up of a set of clusters distributed in the sites of the project partners (Callisto and Orion at the Second University of Naples, Powercost at the University of Sannio). The clusters are equipped with the PerfCloud framework that supports the lease of virtual clusters. Moreover, we have implemented a virtual appliance containing a prototype of the SLA Manager service. This service was able to interact with the PerfCloud middleware through an intermediate interface that represents the first specification of the RQM.

In the role of C@H Admin, we first have set up the system as specified in subsection IV-A, building up a VM on the PerfCloud CP with all the C@H IaaS components. Then, we have started an SLA negotiation process as described in Figure 4, which ended with the creation of an SLA and the deployment of a VC on the PerfCloud CP. After that, we have simulated a fault on a VC node, verifying that the recovery process was triggered and that a new VC node was successfully created and integrated into the VC.

Assuming that MAGDA agents report every 5 minutes (in order to keep the overhead low) and that other 5 minutes are necessary to start a new node and to integrate it in the faulty VC (this is perfectly reasonable, according to our measurements), every fault leads to less than 10 minutes unavailability. In the hypothesis of 4 outages a day (for the same VC, which is though unlikely), we would have an overall downtime of 40 minutes a day, and thus an availability of about 97%.

V. Related work

To the best of our knowledge none of the main commercial IaaS providers (Amazon, Rackspace, GoGRID, ...) offers negotiable Service Level Agreement. What they usually propose is an SLA contract that specifies simple grants on uptime ratio or network availability. Moreover, most of the providers offer additional services (e.g., Amazon's CloudWatch) that monitor the state of target resources (i.e., internal/external bandwidth). Open cloud engine software as Eucalyptus, Nimbus, openNebula, also implements monitoring services for the private cloud provider, but does not provide solutions for SLA negotiation and enforcement. A survey of the SLAs offered by commercial cloud providers can be found in [14]. In [15] the authors describe a system able to combine SLA-based resource negotiations with virtualized resources, pointing out how current literature is still lacking in approaches which take into account both these aspects. A proposal for a global infrastructure aiming at offering SLA on any kind of Service Oriented Infrastructure (SOI) is the objective of the SLA@SOI project [10], which proposes a general architecture that can be integrated in many existing solutions. Anyway, such an interesting solution could be difficult to be used in an infrastructure composed of unreliable resources such as the one targeted by the Cloud@Home project. The Cloud@Home SLA Module aims at offering a set of solutions which enable the C@H Provider to offer SLA services on the top of the resource/Cloud providers that do not offer specific monitoring and SLA features.

Volunteer and Cloud Computing integration. In [16] the authors present the idea of leveraging volunteered resources to build a form of dispersed clouds, or *nebulas* as they call them. Nebulas are intended to complement the offering of traditional homogeneous clouds in some areas where a more flexible can be beneficial, as in testing environments or in application where data are intrinsically dispersed and centralizing them would be costly. BoincVM[17] is an integrated cloud computing platform that can be used to harness volunteer computing resources as laptops, desktops and server farms, for computing CPU intensive scientific applications. It leverages on existing technologies like the BOINC platform and VirtualBox along with some projects currently under development: VMWrapper, VMController and CernVM. Thus, it is a kind of volunteer-on-cloud approach, whereas C@H can be classified as cloud-on-volunteer system.

Resource provisioning from multiple clouds. Even if cloud computing is still an emerging field, the need of moving out from the limitations of provisioning from a single provider is gaining growing interest both in academic and commercial research. In [18], the authors move from a data center model (in which clusters of machines are dedicated to running cloud infrastructure software) to an ad-hoc model for building clouds. The proposed architecture aims at providing management components to harvest resources from non-dedicated machines already available within an enterprise. The need for intermediary components (cloud coordinators, brokers, exchange) is explained in [19], where the authors outline an architecture for a federated network of clouds (the InterCloud). The evaluation is conducted on a simulated environment modeled through the CloudSim framework, showing significant improvements in average turnaround time in some test scenarios.

VI. Conclusions and future work

The ideas described in this paper stem from the proposal of building a cloud infrastructure from volunteered resources (i.e., resources that are voluntarily shared by their owners). By its nature such a cloud will never be able to provide reliable services to consumers, as availability of the resources can not be guaranteed (resources unpredictably join and leave). What we have proposed is the design and implementation of a system that aggregates volunteered resources, and provides on top of them an extra layer that takes care of guaranteeing high levels of resource availability. The system is provided with functionality and services to negotiate the desired level of QoS in the form of resource availability, to monitor the actual service provision level and to carry out actions to guarantee the contracted QoS level against faults due to unexpected unavailability of the underlying resources. We have also described some use case scenarios showing the dynamics of the implemented system.

In this paper the focus has been on a specific scenario where a single volunteer cloud provider is willing to offer its computational resources (in the form of virtual clusters). In our future work, we will investigate on how to provide guarantees on a broader set of quality parameters, other than resource availability. Furthermore, we are planning to give evidence of the viability of the proposed approach also in scenarios where resources are gathered from multiple volunteer providers.

References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE'08*. Ieee, 2008, pp. 1–10.

[2] Peter Mell and Tim Grance, "The nist definition of cloud computing," http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc, 2009.

[3] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009, pp. 124–131.

[4] Distributed Systems Architecture Research Group, "OpenNebula Project," Universidad Complutense de Madrid, 2009, http://www.opennebula.org/.

[5] U. of Chicago, "Nimbus project," 2009, http://workspace.globus.org/clouds/nimbus.html.

[6] V. Casola, A. Cuomo, M. Rak, and U. Villano, "The *Cloud-Grid* Approach: Security and Performance Analysis and Evaluation," *Future Generation of Computer Systems*, 2011.

[7] F. Tusa, M. Paone, M. Villari, and A. Puliafito, "Clever: A cloud-enabled virtual environment," in *Computers and Communications (ISCC), 2010 IEEE Symposium on*, june 2010, pp. 477 –482.

[8] "CloudHome project," 2010, https://cloudathome.unime.it/.

[9] A. Cuomo, G. Di Modica, S. Distefano, M. Rak, and A. Vecchio, "The Cloud@Home architecture: Building a cloud infrastructure from volunteered resources," in *Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER 2011)*, 7-9th May 2011, pp. –, to appear.

[10] "SLA@SOI project: IST- 216556; empowering the service economy with sla-aware infrastructures," http://www.sla-at-soi.eu/.

[11] A. Cuomo, M. Rak, and U. Villano, "Chase: an autonomic service engine for cloud environments," in *Proc. of WETICE 2011, Paris (FR), June 2011*, 2011, to appear.

[12] R. Aversa, B. Di Martino, N. Mazzocca, and S. Venticinque, "Magda: A mobile agent based grid architecture," *Journal of Grid Computing*, vol. 4, no. 4, pp. 395–412, 2006.

[13] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web services agreement specification (ws-agreement)," in *Global Grid Forum*. The Global Grid Forum (GGF), 2004.

[14] L. Wu and R. Buyya, "Service Level Agreement (SLA) in Utility Computing Systems," *Arxiv preprint arXiv:1010.2881*, 2010.

[15] A. Kertesz, G. Kecskemeti, and I. Brandic, "An SLA-based resource virtualization approach for on-demand service provision," in *Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing*. ACM, 2009, pp. 27–34.

[16] A. Chandra and J. Weissman, "Nebulas: using distributed voluntary resources to build clouds," in *Proceedings of the 2009 conference on Hot topics in cloud computing*. USENIX Association, 2009, pp. 2–2.

[17] B. Segal, P. Buncic, D. Quintas, D. Gonzalez, A. Harutyunyan, J. Rantala, and D. Weir, "Building a volunteer cloud," *Memorias de la ULA*, 2009.

[18] G. Kirby, A. Dearle, A. Macdonald, and A. Fernandes, "An Approach to Ad hoc Cloud Computing," *Arxiv preprint arXiv:1002.4738*, 2010.

[19] R. Buyya, R. Ranjan, and R. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," *Algorithms and Architectures for Parallel Processing*, pp. 13–31, 2010.