THE CLOUD@HOME ARCHITECTURE
Building a Cloud Infrastructure from Volunteered Resources

Antonio Cuomo
Dip. di Ingegneria, Università degli Studi del Sannio, Benevento, Italy

Giuseppe Di Modica
Dip. di Ingegneria Elettrica, Elettronica ed Informatica, Università di Catania, Catania, Italy

Salvatore Distefano
Dip. di Matematica, Università di Messina, Messina, Italy

Massimiliano Rak
Dip. di Ingegneria dell'Informazione, Seconda Università di Napoli, Aversa, Italy

Alessio Vecchio
Dip. di Ingegneria dell'Informazione, Università di Pisa, Pisa, Italy

Keywords: Cloud computing, Volunteer computing, IAAS, SLA, QoS, Sensor, Mobile agent.

Abstract: Ideas coming from volunteer computing can be borrowed and incorporated into the Cloud computing model. The result is a volunteer Cloud where the infrastructure is obtained by merging heterogeneous resources offered by different domains and/or providers such as other Clouds, Grid farms, clusters, and datacenters, till single desktops. This new paradigm maintains the benefits of Cloud computing (such as service oriented interfaces, dynamic service provisioning, guaranteed QoS) as well as those of volunteer computing (such as usage of idle resources and reduced costs of operation). This paper describes the architecture of Cloud@Home, a system that mixes both worlds by providing mechanisms for aggregating, enrolling, and managing the resources, and that takes into account SLA and QoS requirements.

1 INTRODUCTION

The core idea of this work is the implementation of a volunteer Cloud, where the infrastructure is built on top of resources voluntarily shared (for free or by charge) by their owners or administrators (*Cloud Providers* or CPs) following a volunteer computing approach. Since this new paradigm merges Volunteer and Cloud computing goals, it is named *Cloud@Home* (C@H for short). It can be considered a generalization and a maturation of the *@home* philosophy, knocking down the (hardware and software) barriers of Volunteer computing by allowing the providers to share more general services. According to this new paradigm, the user resources are not only a passive interface to Cloud-based services, but can actively contribute to create a larger infrastructure. To make this possible the different providers must be able to interoperate with each other. The Cloud@Home paradigm could be also applied to commercial Clouds, establishing an *open computing-utility market* where users can both buy and sell their resources as services. This can be done at different scales: from the single contributing user, who shares his/her desktop, to research groups, public administrations, social communities, small and medium enterprises, which share their computing resources with the Cloud.

Cuomo A., Di Modica G., Distefano S., Rak M. and Vecchio A..
THE CLOUD@HOME ARCHITECTURE - Building a Cloud Infrastructure from Volunteered Resources.
DOI: 10.5220/0003391404240430
In *Proceedings of the 1st International Conference on Cloud Computing and Services Science* (CLOSER-2011), pages 424-430
ISBN: 978-989-8425-52-2

The main goals of Cloud@Home are:

- a) **Volunteer Cloud.** Cloud@Home aims at providing mechanisms and tools for the management of the resources contributed by different providers. This is useful to reconvert the investments made on Grid computing and similar distributed systems into Cloud computing infrastructure, and/or to combine internal resources with external resources for managing peaks or bursts of workload;
- b) **Interoperability among Clouds.** Implementation of mechanisms and tools for achieving interoperability among different Clouds. In this way, any Cloud, either commercial as Amazon EC2 or Azure, or open, based on Eucalyptus, Nimbus and similar, can share resources in C@H, thus becoming a CP for the latter. This will allow to implement a large cross-Cloud infrastructure able, for example, to reduce costs by mixing free volunteer CPs with commercial CPs;
- c) **Integration of Sensing Elements.** Definition and implementation of mechanisms and tools for building a Cloud where sensing elements can participate in a *Infrastructure-as-a-Service* (IaaS) perspective. The set of virtualizable resources can be extended, for instance, to the sensing peripherals available on mobile devices. In this way, the sensing infrastructure can be virtualized and offered on a volunteer-based approach. The global result is a sort of virtualized computer where processing, storage, and input/output (in the form of sensing peripherals) are all composed through the aggregation of heterogeneous resources. To the best of our knowledge this is one of the first attempts to actively involve sensors into Cloud infrastructures.

The final aim of the Cloud@Home project is therefore the design and implementation of a software substrate able to manage such a complex infrastructure and achieve the previously mentioned goals. This requires to face several sub-problems that include: *i*) aggregation of heterogeneous resources; *ii*) monitoring and managing of resources according to specified QoS requirements; *iii*) guaranteing the quality offered by volunteers through the use of flexible Service Level Agreements (SLAs); *iv*) offering the aggregated resources in a service-oriented perspective, as typical of the Utility/Cloud computing approach.

The remainder of the paper is organized as follows: Section 2 introduces the C@H architecture; Sections 3, 4 and 5 detail the specific parts of such architecture; Section 6 draws the conclusions.

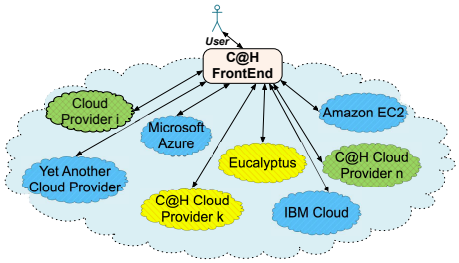

Figure 1: Resource aggregation from many Cloud Providers.

2 OVERVIEW

Cloud@Home aims at collecting infrastructure resources (computing power, storage and sensor) from many different resource providers and offering them through a uniform interface, under an IaaS perspective. As depicted in Figure 1 the resources to be offered are gathered from different providers, which can vary from typical Cloud Providers to PCs hosting a dedicated software or smartphones offering their sensors.

We define *Resource Owner* any private or public subject that owns resources (basically, hardware systems such as computational and storage elements, or sensing devices such as smartphones) and that allows Cloud@Home to access them. In the context of Cloud@Home, the role of Cloud Provider is extended to also include the *Resource Provider*, who may offer many different types of resources using an *as-a-Service* approach. The main goal of Cloud@Home is to provide a set of tools to build up a new, enhanced provider of resources (namely, a *C@H Provider*) who is not yet another classic Cloud Provider, but rather acts as a resource aggregator: it collects resources from many different C@H Providers, who rely on different technologies and adopt heterogeneous resource management policies, and offers such resources to the users. As an example, the *C@H Provider* aims at aggregating virtual machines obtained from a commercial Cloud Provider like Amazon EC2 and virtual machines that run on the PCs of a university lab.

According to this approach we identify two Cloud@Home actors, with the following roles:

C@H User: interacts with the *C@H Provider* in order to obtain the resources. C@H Users are unaware of the nature of the obtained resources. From the C@H Users point of view, the heterogeneity of the Cloud Providers from which resources are collected is completely hidden: they simply demand resources from a *C@H Provider*, which in turn is in charge of

collecting and delivering the needed (virtualized) resources.

C@H Admin: builds up and manage 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.

The C@H Provider offers, along with basic services dedicated to the collection of volunteer-based and commercial resources, a set of tools and services for the Service Level Agreement (SLA) management, monitoring and enforcement. The whole set of functionalities supplied by the C@H Provider is organized into three modules. The **Resource Abstraction Module** hides the heterogeneity of resources collected from Cloud Providers (computing/storage elements, sensor devices) and offer the C@H User a uniform way to access them. The module provides a layer of resource virtualization which abstracts the resource from the specific provider it is taken from. Thanks to specific virtualization functionalities, resources can be received from personal computers (in terms of computing power and storage) and from mobile handheld devices (in terms of sensed data). On top of the Resource Abstraction Module, C@H provides tools for the management of the QoS to be delivered to the C@H Users. Formal guarantees on the performance of resources are offered through the **SLA Management Module**. C@H Users are given the possibility to negotiate the quality level of the requested resources, and are offered contracts (SLAs) that provide guarantees on the resource provision. The negotiation process relies on a performance prediction tool to assess the sustainability of users' requests. A mobile agent-based monitoring function can also be activated to collect information on the actual performance of the delivered resources. The **Resource Management Module** is responsible for the provision of resources. A Resource Discovery tool collects and indexes the resources available for the C@H Users. The enforcement of SLAs is carried out by the Resource and QoS Manager, which basically takes care of resource management (allocation, de-activation, re-allocation) according to the SLA goals. A Frontend acts as an interface that, on the one end gathers the users' requests, and on the other dispatches them to the appropriate system module.

In Figure 2 the C@H system architecture is introduced. In the topmost part of the figure the main components of the C@H IaaS are shown. At the bottom, the resource providers are also shown. The modules that have been described so far are implemented in terms of *C@H Components*. C@H Components are able to interact with each other through standardized, service-oriented interfaces. Such a choice enables flexible component deploying strategies. Pushing the virtualization paradigm to its limits, a single C@H Component can even be offered as a customized virtual machine hosted by any Cloud Provider. The presented approach can be described as a *PaaS* Cloud system which offers basic components in order to build up a Cloud@Home Provider according to the *IaaS* model.

3 RESOURCE ABSTRACTION

Cloud@Home classifies the resources in terms of the kind of resource managed or the resource classes. At the state of the art Cloud@Home supports two different resource classes: *Computational* resources, that are managed in terms of Virtual Machines and/or Virtual Clusters, and *Sensor* resources, that are managed by a service-based interface.

Cloud@Home mainly acts as an intermediary, acquiring resources from different resource providers (eventually Cloud Providers) and delivering them to users. Cloud@Home provides two acquisition modalities for obtaining resources from Cloud Providers: *charged* if they are obtained from commercial Cloud Providers for charge; *open* if obtained from Cloud Providers delivering resources for free (like academic clusters, eventually subjected to internal administration policies and restrictions); *volunteer* if obtained from *Resource Owners* which voluntarily support Cloud@Home (like a laboratory providing machines out of working time). Cloud@Home offers different ways of setting up appropriate Providers for the computing resource classes. To enable the integration of Charged Resource Providers, Cloud@Home

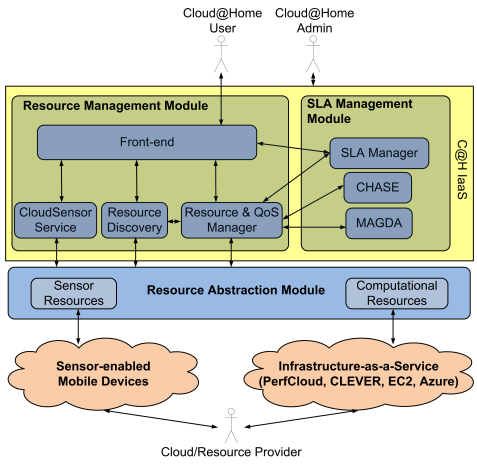

Figure 2: The C@H system architecture.

can interact with some of the main commercial Cloud platforms, like Amazon EC2 and Microsoft Azure. Open and Volunteer computing resources are supported through specific frameworks.

In order to implement a first prototype of Cloud@Home, we choose to adopt two distinct frameworks, namely the *PerfCloud* (Mancini et al., 2009) and *CLEVER* (Tusa et al., 2010). PerfCloud addresses the problem of transforming typical academic facilities like clusters or GRIDs into Cloud Resource Providers, so it is suited to provide the delivery of Open Computational Resources. CLEVER aims at providing Virtual Infrastructure Management services and suitable interfaces to enable the integration of high-level features such as Public Cloud Interfaces, Contextualization, Security and Dynamic Resources provisioning, adopting a strongly decentralized approach, leveraging on P2P protocols and fault-tolerance mechanisms in order to support host volatility.

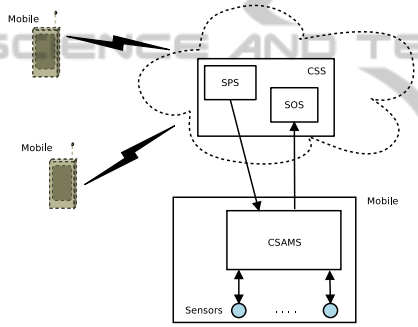

Figure 3: Architecture of the CloudSensor subsystem.

CloudSensor is the subsystem of Cloud@Home that defines the mechanisms enabling the integration of the sensors commonly available on smartphones, such as the accelerometer, GPS, microphone, etc., within the Cloud infrastructure. In this way, applications and services will be able to access a possibly large set of heterogeneous sensors through a service-based interface. CloudSensor (Figure 3) is composed of two subsystems: the *CloudSensor Service* (CSS) and the *CloudSensor Acquisition and Management System* (CSAMS). CSS is hosted within the "fixed" cloud infrastructure and receives the acquisition requests coming from client applications (i.e, the applications that make use of sensing information). Requests are forwarded to CSAMS, executed on the mobile device, that provides the sensor resource abstraction. CSAMS is in charge of receiving the requests coming from the fixed Cloud, evaluating the feasibility of the request (with respect to the user's preferences), carrying out the sensing activity, buffering and then transferring the result to the Cloud.

4 RESOURCE MANAGEMENT

Cloud@Home providers can manage different kind of resources, mainly acting as just intermediaries i.e., delegating the management to resource owners or Cloud Providers. Thus Cloud@Home providers search for cloud/resource providers and, on behalf of the final C@H Users, acquire resources that eventually will be delivered to them.

The *Frontend* component has two main roles in the Cloud@Home architecture: i) it acts as an *access point* for the C@H Users in order to let them acquire the resources; ii) it provides the reference links to all the Cloud@Home components, thus operating as a *glue* entity for the Cloud@Home architecture. In order to provide a user-friendly and intuitive interface to the system, the external side of the Frontend will be developed as an extensible Web Portal that can be also deployed into a virtual machine, in order to simplify the infrastructure access point setup for the C@H Admin. Internally, the C@H Frontend hosts the clients for all the other C@H components: these clients can be easily configured by uploading a configuration file containing the references (URL, IP address, WS Endpoint Reference) to all the active components. The *C@H Web portal* side of the Frontend contains some simple predefined Web modules able to start the C@H components clients and invoke the C@H services.

The *Resource & QoS Manager (RQM)* component implements the (physical and virtual) resources management in C@H. It is responsible for the management of the Cloud resources and services needed to achieve the application requirements imposed by the SLA manager. This task is therefore performed by also taking into account QoS requirements specified during the SLA negotiation process with the C@H User. The RQM has also an important role during the negotiation phase whose responsible is the SLA manager. During the negotiation it identifies a possible resources configuration by querying the Resource Discovery component which, in collaboration with CHASE, will provide an estimation of the performance that can be guaranteed by such a configuration. Other important RQM functionalities are the management of virtual machines and virtual clusters (creation/destruction/migration) and the coordination of monitoring activities, according to the QoS requirements to satisfy. The QoS monitoring is implemented by invoking the service offered by MAGDA, which can deploy an autonomous agent network able to monitor the resources and the QoS they provide. Based on both the monitoring data thus obtained and the information provided by CHASE, the RQM will specify the enforcement policy that is able to ensure

the attainment of the requested QoS.

The *Resource Discovery* component carries out the discovery task. It has to manage the dynamics of the CPs that voluntarily contribute to the C@H by providing their physical resources, since they can asynchronously join and leave the infrastructure, without any notification. It therefore has to implement mechanisms and tools for the voluntarily resources/CP enrolment and management. The discovery tool manages the list of the registered resource/cloud providers, as well as that of the kind and amount of resources they are willing to offer C@H.

The *CloudSensor Service* accepts the sensing requests coming from the client applications, dispatches the requests to the relevant mobile devices, and returns the results to the client applications. CSS is made accessible to the client applications by means of the Frontend. CSS is designed according to the guidelines of the Sensor Web Enablement (SWE) architecture (Botts et al., 2008), specified by the Open Geospatial Consortium, and it is composed of two subsystems. A first service, implemented as an instance of the Sensor Planning Service (SPS) of the SWE architecture, receives the sensing requests from client applications. This service acts as a broker between client applications and the set of mobile devices able to perform the sensing activity. A second service, implemented as an instance of the Sensor Observation Service (SOS) of the SWE architecture, provides an interface useful to request, filter and retrieve the results of the sensing activity. It acts as a decoupling element thanks to its storage facility.

5 SERVICE LEVEL AGREEMENT MANAGEMENT

One of the most important problem to consider in Cloud contexts is the Service Level Agreement management. The volunteer contribution in C@H dramatically complicates the SLA management task, and therefore new strategies must be adopted to satisfy the QoS requirements. In C@H the SLA management is implemented through the interaction of three components: the SLA Manager, the CHASE and the MAGDA, briefly described in the following.

The *SLA Manager* component aggregates and offers C@H Users the templates of the SLAs published by the Cloud Providers that the C@H interacts with. The C@H User can refer to these templates to start a negotiation procedure with the SLA Manager, that will eventually produce an SLA. Since resources (which are the object of the SLA) are virtualized, and can be provided by several, different Cloud Providers adopting different management policies, we argue that SLAs are a fundamental means to guarantee the QoS in such a heterogeneous environment. A rigid vision of the SLA is not suitable to this context. In order to face possible fluctuations of the resources' availability, and to best satisfy the dynamic need and expectations of the C@H users, the SLA must be flexible and adaptable to the context (DiModica et al., 2007). The SLA Manager will adopt a new flexible scheme of negotiation and re-negotiation of the QoS. According to this scheme, whenever violations on the SLA's guarantees are about to occur, run-time (i.e., during the service provisioning) re-negotiations and modifications of the guarantees are allowed to take place. On a successful re-negotiation, the SLA would be accordingly modified and the QoS level is adjusted; should the negotiation fail, the SLA would not get modified and would still apply on the resource provisioning. In the context of voluntarily offered resources, resources providers can deliberately decide to share or un-share their resources at any time, the fluctuation of the resources' availability is a big issue to face with. Flexible SLAs aim at preserving the continuity of the resource provisioning, which for sure is a goal for both the C@H Users and the provider of resources. Finally, when the agreement has been reached (i.e., both the end user and the C@H provider agree on the SLA terms), the SLA is passed to the RQM, who is responsible for its enforcement. The SLA Manager will also provide the end user with an end point reference through which the SLA can be monitored.

CHASE is the component in charge of delivering reliable predictions on the performance of the applications running in the Cloud. It evolves from an existing framework for the autonomic performance management of service oriented architectures (Casola et al., 2007). CHASE main components are an optimization engine and a trace-driven discrete event simulator. Applications can be modeled through a formalized description of the application components, for which an XML based language, MetaPL, can be used. In the context of this project, CHASE plays multiple roles. Firstly, it is invoked during the service negotiation phase: when the SLA Manager needs to evaluate the sustainability of user requests, it transmits to the RQM component a formalized description of the QoS Level explicitly requested by the user. The RQM forwards this description to the CHASE simulator, along with an application description and information about the current state of system, in terms of resource availability and load. CHASE uses the application description to build traces that can be fed to the simulator, the status information to update the

platform description and the QoS description to build a parametrized objective function to be minimized (or maximized, depending on the nature of the parameters). The optimization engine drives the simulator to explore the space of possible configurations and informs the RQM whether a configuration meeting the demands has been found or not. CHASE can also be invoked by the RQM component in case that the monitoring system is alerting that the QoS agreed in the SLA is at risk of violation: by performing new simulations in the up-to-date setting, it informs the RQM if alternative scheduling decisions (like migrating or adding more VMs) can solve the problem or if the violation is unavoidable, in which case the SLA will have to be renegotiated or terminated.

The *MAGDA* component is a service hosted into a virtual machine and implementing the MAGDA (Aversa et al., 2007) mobile agent platform (a mobile agent is a Software Agent with an added feature: the capability to migrate across the network together with its own code and execution state, following both pull and a push execution model (Xu and Wims, 2000)). It is developed using and extending the JADE, a FIPA compliant agent platform developed by TILAB (Bellifemine et al., 2001). The *MAGDA* Platform hosts a set of agents able to perform different kind of benchmarks, that vary from simple local data sampling (actual CPU usage, memory available, . . .) to distributed benchmarking (evaluating distributed data collection, or evaluating the global state with snapshot algorithms). Moreover the mobile agents are able to update an archive of the measurement in order to perform historical data analysis. In the Cloud@Home context, the migration capabilities are of help to carry on an on-site (i.e., on the resource itself) monitoring. RQM and SLA-oriented components use the mobile agents in order to monitor the delivered resources. Performance indexes are collected and used to enforce the SLAs. In the case that the C@H User explicitly requests the monitoring service, mobile agents are instructed to migrate to the place where the resources have been activated and start monitoring their performance at application level. It is also possible to customize the monitoring procedure and remotely check the state of the resources, following either a push (agents autonomously provides information on resources) or a pull (agents are polled in order to get information on resources) communication model.

6 CONCLUSIONS

The idea of exploiting unused computing and storage resources is at the base of the Volunteer computing as well, according to which private users' resources are voluntarily aggregated to serve distributed, mostly scientific, applications. This work grounds on the idea of exploiting voluntarily offered resources (data sensed from mobile devices, single desktops, private or public Companies' Data Centers) to build general purpose clouds that will then re-offer the resources on an "as-a-service" basis. The main objective of the work is to propose a framework (Cloud@Home) for the integration of both commercial and volunteer-based clouds, aggregating resources from heterogeneous environment, and offering users extra services to monitor and guarantee the quality of the provided resources. The architecture of Cloud@Home has been described in the paper. In the future, we are planning to integrate the security requirements among the service level objectives to be guaranteed, and to add a module for the C@H Users' billing and accounting management.

ACKNOWLEDGEMENTS

The work described in this paper has been partially supported by the MIUR-PRIN 2008 project "Cloud@Home: a New Enhanced Computing Paradigm".

REFERENCES

Aversa, R., Martino, B. D., Rak, M., and Venticinque, S. (2007). A framework for mobile agent platform performance evaluation. In *WOWMOM*, pages 1–8. IEEE.

Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Jade: a fipa2000 compliant agent development environment. In *Agents*, pages 216–217.

Botts, M., Percivall, G., Reed, C., and Davidson, J. (2008). Ogc sensor web enablement: Overview and high level architecture. In Nittel, S., Labrinidis, A., and Stefanidis, A., editors, *GeoSensor Networks*, volume 4540 of *Lecture Notes in Computer Science*, pages 175–190. Springer Berlin / Heidelberg.

Casola, V., Mancini, E. P., Mazzocca, N., Rak, M., and Villano, U. (2007). Building autonomic and secure service oriented architectures with MAWeS. In *Autonomic and Trusted Computing, Lecture Notes in Computer Science*, volume 4610, pages 82–93, Berlin (DE). Springer-Verlag.

DiModica, G., Regalbuto, V., Tomarchio, O., and Vita, L. (2007). Enabling re-negotiations of sla by extending the ws-agreement specification. *Services Computing, IEEE International Conference on*, 0:248–251.

Mancini, E. P., Rak, M., and Villano, U. (2009). Perfcloud: Grid services for performance-oriented development

of cloud computing applications. In *WETICE*, pages 201–206.

Tusa, F., Paone, M., Villari, M., and Puliafito, A. (2010). Clever: A cloud-enabled virtual environment. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 477 –482.

Xu, C.-Z. and Wims, B. (2000). A mobile agent based push methodology for global parallel computing. *Concurrency - Practice and Experience*, 12(8):705–726.