Volunteer Cloud Architecture Survey

--- Cloud @ Home ---

Volunteer Computing and Desktop Cloud: the Cloud@Home Paradigm (2009)

In this paper [1], they present what they call the Cloud@Home paradigm. This paradigm is characterized by combining the Cloud Computing paradigm and the Volunteer Computing paradigm, all the while following the "@home" philosophy. One of the unique elements of this paradigm is the fact that they propose to make the volunteer and commercial perspective coexists. They identify 6 key challenges that the architecture needs to tackle: Resources and Services management, Frontend, Security, Reliability, Interoperability and Business models. There is a need to propose a mechanism to manage the resources and services offered by the infrastructure, namely Resource and Services management. There is also a need to provide an homogeneous frontend, with a service oriented point of view, which will act as a uniform access point to the end-users.

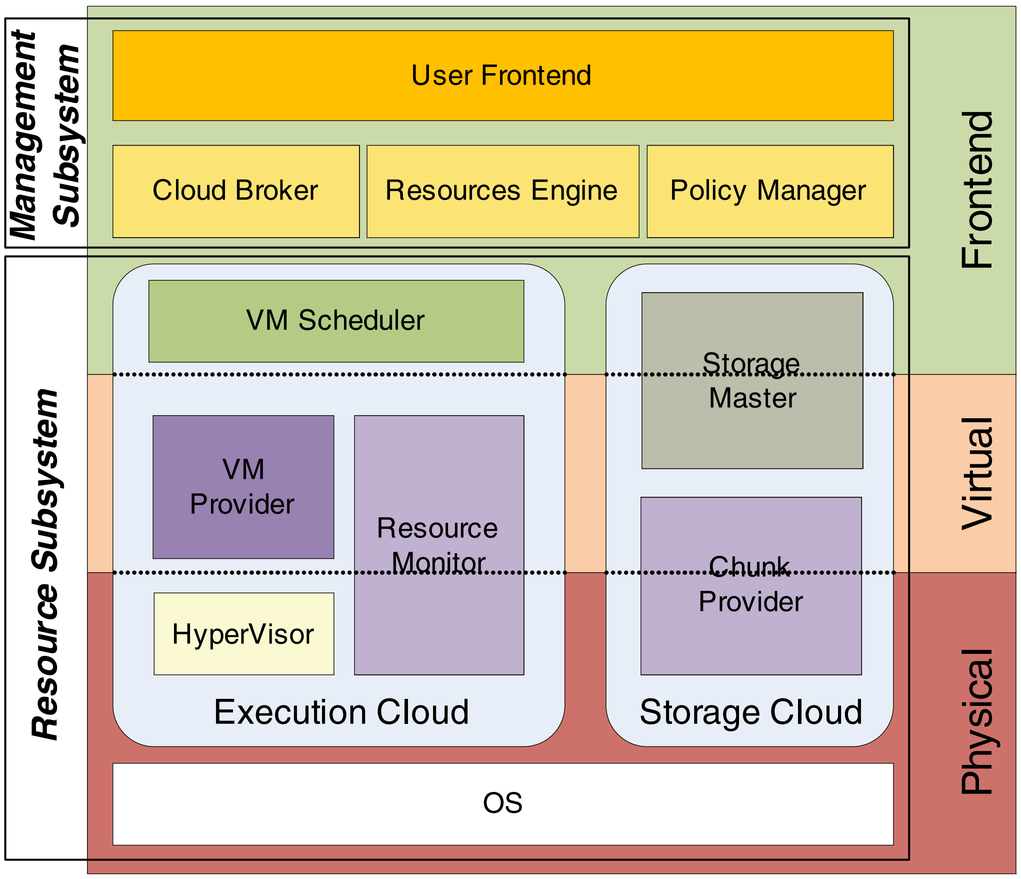

Figure 1 Cloud@Home Core Structure Organization

The figure above represents an initial structure organization of the possible implementation of the Cloud@Home paradigm. This paper really provided a basis on which future Volunteer Cloud Architecture could build upon, by presenting this hybrid model embracing cloud service provider interoperability and volunteering.

The Cloud@Home Architecture (2011)

They propose [5] an architecture to build a volunteer-based cloud environment, and present what they call the Cloud@Home paradigm. They present a vision of using a heterogeneous base of independently provided resources through a common service-based interface. In this paper, they present an architecture that take into account resource discovery, resources management, and resource provisioning and service-level agreement management. One key aspects presented is to incorporate premium services like Amazon EC2 to the resource base, thus always being able to provide access to resources, but charging a price. Also, they propose the idea of integrating sensing elements, and thus providing a sensing infrastructure via different mobile devices providing to the sensing resource base.

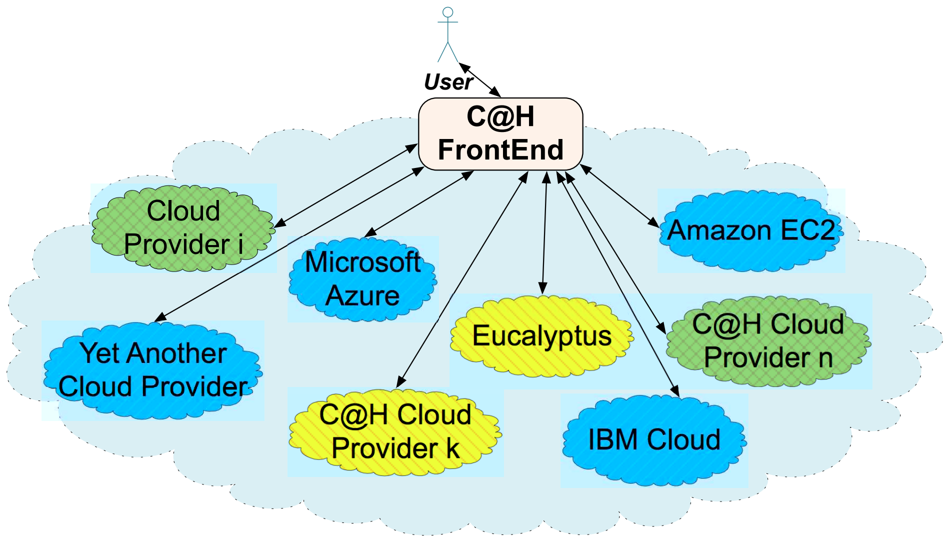

Figure 2 Resource aggregation

The figure above shows the concept of aggregating free and premium cloud resources. The authors define the main goals of C@H as being the following: to provide a set of tools to build up a new, enhanced provider of resources which acts as a resource aggregator, and offer these heterogeneous resources through an homogeneous interface to the end-users. It operates at the IaaS service level, the following figure shows the designed architecture:

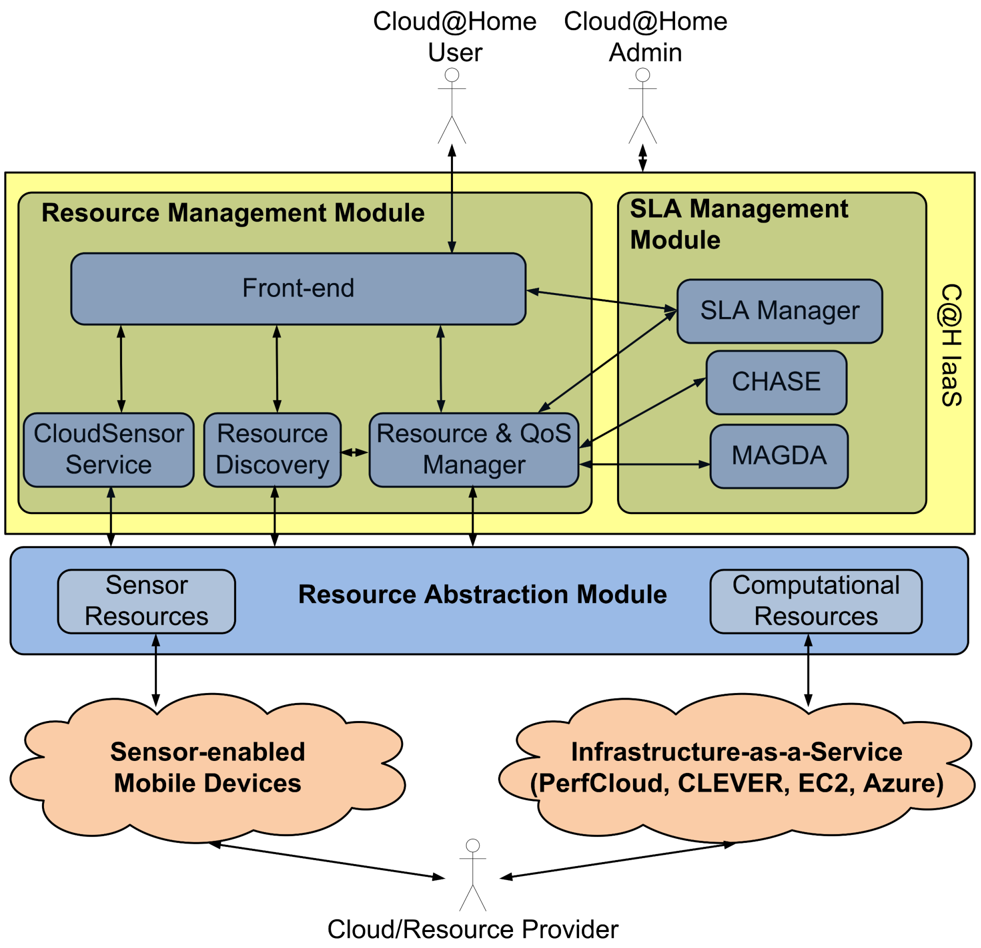

Figure 3 C@H system architecture

The architecture they propose is composed of three modules, namely: the Resource Management Module, the SLA Management Module and the Resource Abstraction Module. They used two frameworks to implement their first prototype, PerfCloud [6] and CLEVER [7]. PerfCloud is concern with refactoring GRIDs and clusters into Cloud Resource Providers. CLEVER, on the other hand is concerned with Virtual Infrastructure Management services, and enable the integration of high-level features in the likes of Public Cloud Interfaces, Security and much more. It also enables the usage of a strongly decentralized approach by leveraging P2P protocols.

--- Nebulas ---

Nebulas: Using Distributed Voluntary Resources to Build Clouds

In this paper [8], the authors present three classes of services for which traditional commercial cloud service providers may be inadequate. They are inadequate in the sense that they are not able to provide these services for a reasonable financial compensation. These services are the following: Experimental cloud services, Dispersed-Data-intensive services and Shared Services. The first one is concerned with providing access to a large-scale infrastructure to test-drive potential cloud application to assess the ability to be a cloud service. Also, in order to make it production-read, currently there is no way to test-drive a potential cloud service before adhering to any cloud service provider and paying a hefty premium. The second one is concerned with services that rely on large amounts of dispersed data and to move all the data to a centralized cloud would be to expensive or inefficient. Rather it would be advantageous to move the computational resources closer to where the data reside, thus reducing the bandwidth requirements and the space needed to centralize the data to be analyzed. And the final one is concerned with services to provide freely a private application to others as a public service, but requires deployment resources. Also there might be a need to scale-up and scale-down based on the demand.

From this birth the concept of *nebulas*: more dispersed, less managed clouds, constructed using voluntary resources (donated by end-user hosts similar to the C@Home systems). They are complimentary to commercial clouds and in some case would lead to a migration to commercial clouds.

Early Experience with the Distributed Nebula Cloud (2011)

In this paper [9], the authors experiment with *Nebulas: a dispersed, context-aware, and cost-effective cloud.* They define Nebulas has having highly dispersed set of computational and storage resources, so that it can exploit locality to data sources and knowledge about user context more easily. It will include edge machines, devices and data sources as its resources, and most of the resources would be autonomous and loosely managed, reducing the need for costly administration and maintenance, thus a cost-effective solution. One of the important characteristics that the authors put emphasis upon, is the fact that the Nebulas are not proposed to replace Commercial Clouds, but rather compliment them.

The focus is put on one of the potential services presented in [8], namely dispersed data-intensive services. They present their early experience with a prototype implementation of a Nebula based on the use of volunteer dispersed resources. The results aren't very interesting for the purposes of this survey.

They present the following architecture for the Nebula system, which is composed of three components: a distributed DataStore, a network dashboard and a secure NodeGroup. From a high-level perspective, it is a pool of distributed resources controlled by the Nebula Central. It is a central control node that manages the participant nodes and the applications deployed. The propose of grouping the user-provided resources into two abstractions: a DataStore and a secure NodeGroup.

The DataStore provides basic storage capabilities for application data as well as Nebula specific system/management data.

The NodeGroup represent a set of computational nodes to applications and services. The computational node carries out the actual application computations and interacts with the Nebula Central to register itself as a part or the Nebula. It relies on the Nebula Central for information regarding pending tasks and DataStore nodes that are in proximity. It, similarly to DataStore, consists of voluntary resources that join and leave on an ad-hoc basis. The authors used Google Native Client, which is a browser plugin that enables sandboxing for running native code in Web applications and thus achieving portability, to implement secure computing node.

The DashBoard is a set of networking monitoring tools deployed across all participating nodes.

As a retrospective, the application that they present in this paper is closer to volunteer distributed computing in a de-centralized architecture, rather than Volunteer Cloud Computing. This is mainly attributable to the fact that the computations are de-coupled from the storage, and uses a geography/locality aware work distribution strategy.

--- Miscellaneous Volunteer Cloud Architecture Proposals ---

Towards A Volunteer Cloud Architecture (2013)

In this paper [10, 11], the authors present a very crude possible architecture for Volunteer Cloud Architecture. It is composed of three layers: a service layer, a middleware layer and a physical layer.

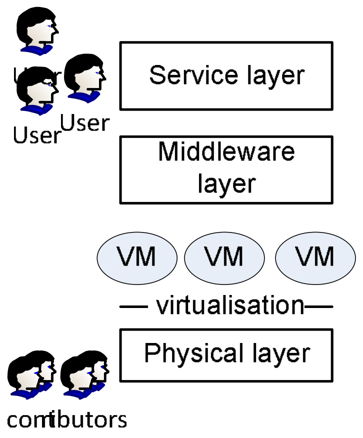

Figure 4 VCC proposed Architecture

The service layer provides services via an interface, based on a SOA approach, to the end-users. VCC's contributors volunteer their resources to form a VCC for a certain time, and they may be services consumers at the same time. The Middleware layer is composed of two components: Task Management, and QoS Management. The first one is concerned with balancing the repartition of the workload and coordinates the distributing of the tasks. The latter one is concerned with providing a minimum quality level. And finally the Physical Layer is concerned with resource aggregation, resource allocation and monitoring.

The author present a plausible conceptual architecture for VCC, no implementation was done to assess practical applicability of the architecture.

--- P2P Volunteer Cloud Architecture Proposals ---

Design and Implementation of a P2P Cloud System (2012)

In this paper [12], the authors propose and describes the design and prototype implementation of a fully decentralized, P2P Cloud. They present the idea of Peer-to-Peer Cloud Computing, which consist of building a cloud out of independent resources that are opportunistically assembled. It could be built by assembling individual peers without any central monitoring or coordination component. It would provide on-demand scalability, access to computing and storage space with no single point of failure nor central management. Resources are added to the pool of resources simply by installing a software daemon on them. Their proposed implementation is advertised as a fully distributed IaaS Cloud infrastructure.

They differentiate themselves from Cloud@Home by putting emphasis on the fact that it relies on centralized components, while allowing users to contribute, (theoretically it isn't required). Also, their architecture is fully de-centralized and it doesn't require any central bookkeeping service. Finally, they note that there is no known implementation to date of the Cloud@Home proposal.

The System Model they propose consists of nodes, and these nodes join by installing a software daemon. This software daemon presents two interfaces: a user interface and a node-to-node interface. The API that is exposed by the user interface is similar to conventional IaaS Cloud APIs (such as Amazon EC2 or S3). Nodes are managed by their respective owners in which case it offers no QoS guarantee. It goes for applications failures/crashes; the responsibility is reverted to the users (as would be the case for conventional IaaS Clouds).

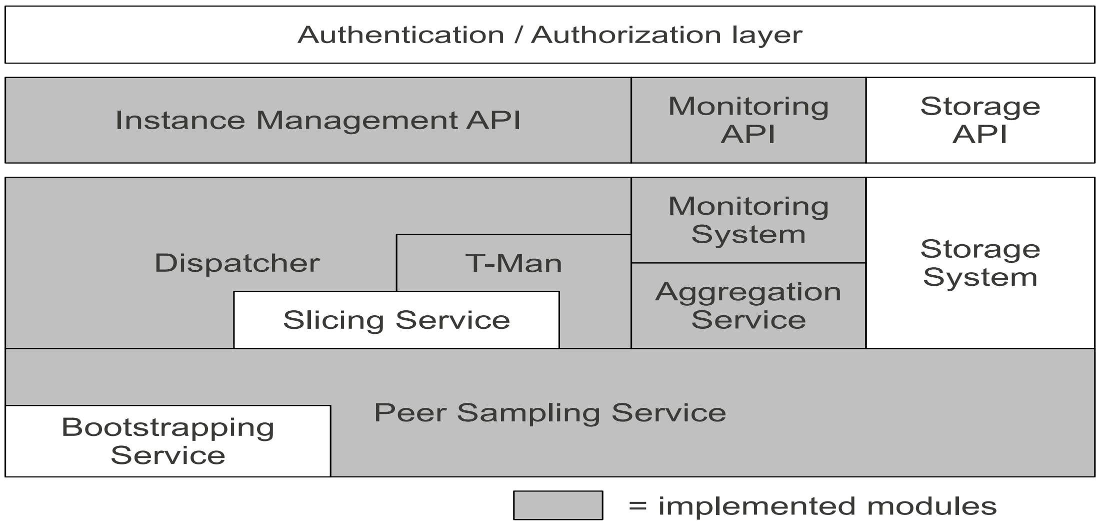

Figure 5 Layered architecture of the P2PCS

The figure above shows the layered architecture they've implemented for the P2PCS. Again, as stated above each node is running a software daemon, which makes for a collection of identical processes running on separate hosts. We will briefly present each of the implemented modules (in gray).

First the Peer Sampling Service (PSS) aims at providing each node with a list of peers to exchange messages with. They achieve this by maintaining an unstructured overlay over the set of peers. It can keep the overlay connected also in presence of churn, by using a simple gossip protocol. It uses a BootStrapping Service to gather an initial set of nodes, although it not implemented yet; the current implementation parses a text file with the IP addresses of the nodes.

Second is the Slicing Service (SS), which is used to rank the nodes according to one or more attributes, it is also used to request slice of the whole cloud according to a user-defined criteria.

Third is the Aggregation Service (AS), which is used to compute global measures using local message exchanges. It allows each peer to know system-wide parameters without the need to access a global registry.

Fourth is the Monitoring System, which is implemented on top of the AS, and it collects global system parameters and then provides them to the user. The MS API provides means to start and stop the display of run-time instance informations. In the current implementation it is used to display the topology of the network and the set of nodes of the slice a node belongs to.

Fifth is the T-Man component, which is used to create a overlay network with a given topology, and it is based on gossip protocols.

Sixth is the Dispatcher, which is responsible for handling the requests submitted by the user through the high level user interface and translate them into the appropriate low level gossip protocol commands which are sent to the other nodes.

Final is the Instance Management API, which contains all the functionalities to create and terminate instances, and also to provide means to list which resources are held by this user.

The implementation was done in Java, using the server-client paradigm. It still resides online at:

http://cloudsystem.googlecode.com/svn/trunk/ source/

It seems to be a dead project, since no updates were done or any changes were made since 2011.

--- Honorable Mentions (Bit off-topic) ---

P2P MapReduce: Parallel data processing in dynamic cloud environment (2011)

--- Thoughts and Conclusions ---

Cloud@Home seems the most extensive project addressing the VCC paradigm. They provide a detailed architecture throughout several publications, but yet still no implementation is known to exist. Also, the only website that was found pertaining to this project contains nothing but agendas and contact info, it hasn't been updated since 2011, it may be safe to assume that this project is stalled. No option is made available as an open-source solution that users can use. Also the fact that it depends on a centralized collection of components makes it different from the P2PCS.

P2PCS is the only known implementation of a P2P Cloud System to date. It was part of thesis, and any development has stop since 2011 (as said above). It offers a decentralized infrastructure and doesn't require any sort of centralized bookkeeping service to manage the current nodes.

Therefore, there is currently a need to further the research in this domain of VCC, more specifically there is a need to propose a truly open-source VCC that can rely on P2P technology, but most importantly that provides a decentralized architecture that embraces the green computing philosophy. Also as part of a free the Internet movement it is imperative to counter the commercial lock-in imposed by the current Cloud Service providers. This can be done by providing a Cloud service, free of charge that is for the community, by the community.

REFERENCES

[1] Cunsolo, V. D., Distefano, S., Puliafito, A., & Scarpa, M. (2009, July). Volunteer computing and desktop cloud: The cloud@ home paradigm. In Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on (pp. 134-139). IEEE.

[5] Cuomo, A., Di Modica, G., Distefano, S., Rak, M., & Vecchio, A. (2011, May). The Cloud@ Home Architecture-Building a Cloud Infrastructure from Volunteered Resources. In CLOSER (pp. 424-430).

[6] Mancini, E. P., Rak, M., & Villano, U. (2009, June). Perfcloud: Grid services for performance-oriented development of cloud computing applications. In Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009. WETICE'09. 18th IEEE International Workshops on (pp. 201-206). IEEE.

[7] Tusa, F., Paone, M., Villari, M., & Puliafito, A. (2010, June). CLEVER: A cloud-enabled virtual environment. In Computers and Communications (ISCC), 2010 IEEE Symposium on (pp. 477-482). IEEE.

[8] Chandra, A., & Weissman, J. (2009, June). Nebulas: Using distributed voluntary resources to build clouds. In Proceedings of the 2009 conference on Hot topics in cloud computing (pp. 2-2). USENIX Association.

[9] Weissman, J. B., Sundarrajan, P., Gupta, A., Ryden, M., Nair, R., & Chandra, A. (2011, June). Early experience with the distributed nebula cloud. In Proceedings of the fourth international workshop on Data-intensive distributed computing (pp. 17-26). ACM.

[10] Alwabel, A., Walters, R., & Wills, G. (2013). Towards a volunteer cloud architecture. In Computer Performance Engineering (pp. 248-251). Springer Berlin Heidelberg.

[11] Alwabel, A., Walters, R., & Wills, G. (2012). Towards Performance Evaluation in Volunteer Clouds. University Halle-Wittenberg Institute of Computer Science.

[12] Babaoglu, O., Marzolla, M., & Tamburini, M. (2012, March). Design and implementation of a p2p cloud system. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (pp. 412-417). ACM.