2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing

Applying software engineering principles for designing Cloud@Home

V. D. Cunsolo, S. Distefano, A. Puliafito, M. Scarpa
Università di Messina, Dipartimento di Matematica
Contrada Papardo, S. Sperone, 98166 Messina, Italy
vdcunsolo{sdistefano, apuliafito, mscarpa}@unime.it

Abstract—**Cloud computing is the "new hot" topic in IT. It combines the maturity of Web technologies (networking, APIs, semantic Web 2.0, languages, protocols and standards such as WSDL, SOAP, REST, WS-BPEL, WS-CDL, IPSEC, etc.), the robustness of geographically distributed computing paradigm (*Network*, *Internet* and *Grid computing*) and self-management capabilities (*Autonomic computing*), with the capacity to manage quality of services by monitoring, metering, quantifying and billing computing resources and costs (*Utility computing*). Those have made possible and cost-effective for businesses, small and large, to completely host data- and application-centers virtually... in the Cloud.**

Our idea of Cloud proposes a new dimension of computing, in which everyone, from single users to communities and enterprises, can, on one hand, share resources and services in a transparent way and, on the other hand, have access to and use such resources and services adaptively to their requirements. Such an enhanced concept of Cloud, enriching the original one with Volunteer computing and interoperability challenges, has been proposed and synthesized in Cloud@Home.

The complex infrastructure implementing Cloud@Home has to be supported by an adequate distributed middleware able to manage it. In order to develop such a complex distributed software, in this paper we apply software engineering principles such as rigor, separation of concerns and modularity. Our idea is, starting from a software engineering approach, to identify and separate concerns and tasks, and then to provide both the software middleware architecture and the hardware infrastructure following the hw/sw co-design technique widely used in embedded systems. In this way we want to primarily identify and specify the Cloud@Home middleware architecture and its deployment into a feasible infrastructure; secondly, we want to propose the development process we follow, based on hardware/software co-design, in distributed computing contexts, demonstrating its effectiveness through Cloud@Home.

Keywords: Cloud computing; cross-platform interoperability; Volunteer computing; Separation of Concerns, HW/SW co-design.

I. Introduction and Motivations

Cloud computing is a new computing paradigm that promises to realize most of people desires. Through Cloud computing small and medium enterprises can avoid investment on building their own data centers, and big enterprises can reduce the operative costs of their own data centers. Private users and common people mainly experience the benefits of Cloud in networking: storing data, emails, photos, etc., exploiting web services, and so on. Cloud computing seems to be the panacea of information technology (IT), since services are always available, unbound and delocalized: they are somewhere in the Cloud and everyone can use them.

The development and the success of Cloud computing is due to the maturity reached by both hardware and software, in particular referring to the virtualization [28], [4] and the Web (Web 2.0 [25], WSDL [7], REST [14], SOAP [29]) technologies. It is conceived by the *service-centric perspective* that is quickly and widely spreading in the IT in terms of *service oriented science* [15], *service computing* [31] and *IT as a service* (ITAAS) [16]. From this perspective, all capabilities and resources of a Cloud (usually geographically distributed) are provided to users *as a service*, to be accessed through the Internet without any specific knowledge of, expertise with, or control over the underlying technology infrastructure that supports them.

Cloud computing offers a user-centric interface that acts as a unique, user friendly, point of access for users' needs and requirements. Moreover, it provides *on-demand service provision*, *QoS guaranteed offer*, and *autonomous system* for managing hardware, software and data transparently to users [30].

A great interest on Cloud computing has been manifested from both academic and private research centers, and numerous projects from industry and academia have been proposed. In commercial contexts, among the others we highlight: Amazon Elastic Compute Cloud [1], IBM's Blue Cloud [18], Sun Microsystems Network.com [24], Microsoft Azure Services Platform [8], Google App Engine [19], Dell Cloud computing solutions [10]. There are also several scientific activities, such as: Reservoir [22], Nimbus-Stratus-Wispy-Kupa [26], Eucalyptus [21] and OpenNEbula [12]. All of them support and provide an on-demand computing paradigm, in the sense that a user submits his/her requests to the Cloud that remotely, in a distributed fashion, processes them and gives back the results. This client-server model well fits aims and scopes of commercial Clouds: the business. But, on the other hand, it represents a restriction for scientific Clouds, that have a view closer to *Volunteer computing*, i.e. of using computers volunteered by their owners as a source of computing power and storage to provide distributed scientific computing [2]. It is behind the *"@home"* philosophy of sharing network connected resources for supporting distributed scientific computing.

We believe the Cloud computing paradigm is applicable also at lower scales, from the single contributing user, that shares his/her desktop, to research groups, public administrations, social communities, small and medium enterprises, which make available their distributed computing resources to the Cloud. Both free sharing and pay-per-use models can be easily adopted in such scenarios. Starting from such considerations, in [9] we propose a more "democratic" form of Cloud computing, in which the computing resources of single users accessing the Cloud can be shared with the others, in order to contribute to the elaboration of complex problems. Since this paradigm is very similar to the Volunteer computing one, we named it *Cloud@Home*. Both hardware and software compatibility limitations of Volunteer computing can be overcome in Cloud@Home through virtualization. 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 services.

In order to achieve such ambitious goals, it is necessary to implement an adequate software architecture, a middleware, managing the underlying complex infrastructure combining heterogeneous resources, softwares and middlewares, taking into account security, interoperability, quality of service and other specific issues and challenges. A way for doing that could be to exploit an approach taken from software engineering in the *software development process*, i.e. of concentrating on principles and on a few representative methods and techniques with broad application in SE, such as [17]: *rigour*, *separation of concerns*, *modularity* and *anticipation of change*. *Rigour* means, in this context, the spelling out soundly and comprehensively of each step and each product (including intermediate ones) of the software process, with as much precision as necessary. Creative processes tend to be imprecise and inaccurate, insofar as they are driven by unstructured inspiration. When rigour is implemented as complete formality, it opens the way to mechanization. *Separation of concerns* means concentrating on one aspect and momentarily disregarding the others. Some of the complexity of software development emanates from the trade-offs and dilemmas posed by different aspects of the problem. Three well-known examples are analysis (the what) vs. design (the how), structure vs. behaviour, and data vs. control flow. One can also separate concerns in time (e.g., by making the task a sequence of component tasks) or in terms of quality (e.g., efficiency vs. adequacy). *Modularity* is possibly the most important case of separation of concerns. *Abstraction* is another important case. *Anticipation of change* means taking into account all along the development process that software must evolve.

The main aim of the present paper is to apply SE principles in the specification of both the middleware and the infrastructure of Cloud@Home. This is an approach frequently used in embedded system hardware/software co-design [3], [27], [20], also applied in other context such as autonomic computing [11], high performance computing [13], and so on. The idea has also been further generalized [5], [23], giving impetus to the development of new techniques as the Unified Systems modeling [6]. This strengthens our beliefs of applying the unified hw/sw co-design approach also to distributed contexts such as the Cloud computing. In fact, we think that the advantages of such technique can be amplified in highly distributed systems due to their great complexity, since in such systems the software design is often harder than the infrastructure design, in particular if specific hardware constrains must be satisfied by the software architecture. By reversing the classical design process starting from the distributed hardware infrastructure and successively considering the software architecture, if possible, the software development process could be significantly simplified as well as the whole hw/sw systems, improving the overall efficiency and reducing costs. In other words, we think that the hw/sw co-design is particularly conceived for distributed systems and we try to demonstrate such statement through Cloud@Home.

In this way, starting from a clear and rigorous separation of concerns, functions and responsibilities of the Cloud@Home paradigm, we firstly specify the middleware implementation and therefore we identify the architecture of the infrastructure, following a reverse hw/sw co-design process. Thus, the paper is organized as follows: in section II an overview of the Cloud@Home concepts, requirements and goals is described, then applying the separation of concerns on them; therefore, the Cloud@Home middleware architecture is specified in section III and finally the Cloud@Home infrastructure architecture is defined in section IV, as the deployment of the middleware architecture before specified. Section V summarizes and closes the paper.

II. Overview and Separation of Concerns

Cloud@Home intends to reuse *"domestic"* computing resources to build voluntary contributors' Clouds that can interoperate each other and with external commercial Clouds, such as Amazon EC^2, IBM Blue Cloud, Microsoft Azure Services Platform, and so on. With Cloud@Home, anyone can experience the power of Cloud computing, both actively providing his/her own resources and services, and passively submitting his/her applications.

In Cloud@Home both the commercial and the volunteer viewpoints coexist: in the former case the end-user orientation of Cloud is extended to a collaborative two-way Cloud in which users can buy and sell their resources and services; in the latter case, the Grid philosophy of few but large computing requests is extended and enhanced to *open* Virtual Organizations. In both cases QoS requirements could be specified, introducing both in the Grid and in the Volunteer philosophy (*best effort*) the concept of quality.

Cloud@Home can be also considered as a generalization and a maturation of the @home philosophy: a context in which users voluntarily share their resources without any compatibility problem. This allows to knock down both hardware (processor bits, endianness, architecture, network) and software (operating systems, libraries, compilers, applications, middlewares) barriers of Grid and Volunteer computing, into a service oriented architecture. On the other hand, Cloud@Home can be considered as the enhancement of the Grid-Utility vision of Cloud computing. In this new paradigm, users' hosts are not passive interfaces to Cloud services, but they can be actively involved in computing. Single nodes and services can be enrolled by the Cloud@Home middleware, in order to build own-private Cloud infrastructures that can (for free or by charge) interact with other Clouds.

The key points of Cloud@Home are on one hand the *volunteer contribution* and on the other the *interoperability* among Clouds. Well-known problems for the parallel, distributed and network computing communities have to be addressed regarding security, QoS, SLA, resource enrollment and management, heterogeneity of hw and sw, virtualization, etc. All of them must be contextualized into an highly dynamic environment in which nodes and resources can frequently change state, making themselves suddenly unavailable.

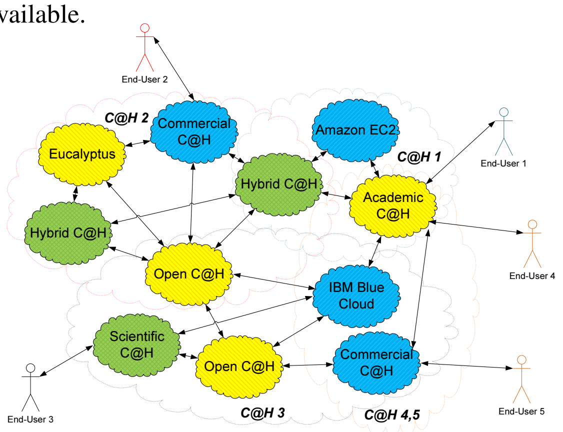

Figure 1. Cloud@Home Scenario

The Cloud@Home point of view is pictorially depicted in Fig. 1, where several different Clouds, also built on volunteered resources (open Clouds), can interact and can provide resources and services to the other federated Clouds.

In order to implement such a form of computing we choose to logically classify concepts, requirements and needs into the model shown in Fig. 2. The separation of concerns principle has been applied there to Cloud@Home, identifying three main different and separated layers: *frontend*, *virtual* and *physical*. Each of them groups all the specific functions, tasks and relationships the middleware have to provide according to the considered perspective. In this way we obtain a *layered model*: the elements of a layer can communicate and cooperate each other, moreover, they provide services to the above layer, and use services delivered by the layer below. Each layer has the knowledge of exclusively its neighbors layers, but does not know nothing about the others.

Coming into details of the Cloud@Home characterization of Fig. 2, as above stated, the separation has been there operated by considering three different viewpoints:

- The *Frontend Layer* that globally manages resources and services (coordination, discovery, enrollment, etc), implements the user interface for accessing the Cloud (ensuring security reliability and interoperability), and provides QoS, business models and SLA policies management facilities.
- The *Virtual Layer* that implements a homogeneous view of the distributed Cloud system offered to the higher frontend layer (and therefore to users) in form of two main basic services: the *execution service* that allows to set up a virtual machine, and the *storage service* that implements a distributed storage Cloud to store data and files as a remote disk, locally mounted or accessed via Web.
- The bottom *Physical Layer* that provides both the physical resources for elaborating incoming requests and the software for locally managing such resources.

The frontend layer offers to the user a uniform view of undefined, unbounded and unlimited computing and storage capabilities. This view is implemented by adaptively managing a pool, a farm, a set of finite virtual resources provided by the lower virtual layer. The virtual layer instead virtualizes the underlying physical resources, breaking down the incompatibility barriers. The physical layer locally manages the nodes, without any knowledge that it is part of a distributed infrastructures. Due to the layered model organization, the frontend layer ignore the existence of a physical layer and viceversa, as introduced above.

In Fig. 2, two kind of users are also distinguished: *contributing hosts* that share their resources in order to compose the Cloud, and *consumer hosts* that can interact with the Cloud submitting their requests after authenticating into the system. One of the main enhancement of Cloud@Home is that a host can be at the same time both contributing and consumer host, establishing a symbiotic mutual interaction with the Cloud.

III. The Middleware

Once the Cloud@Home concerns have been identified and separated, it is possible to specify the functions to implement at each layer, thus building the Cloud@Home middleware architecture. More specifically, in order to implement the Cloud@Home computing paradigm the following issues should be taken into consideration:

- Resources and Services management - a mechanism for managing resources of the Clouds is mandatory. This

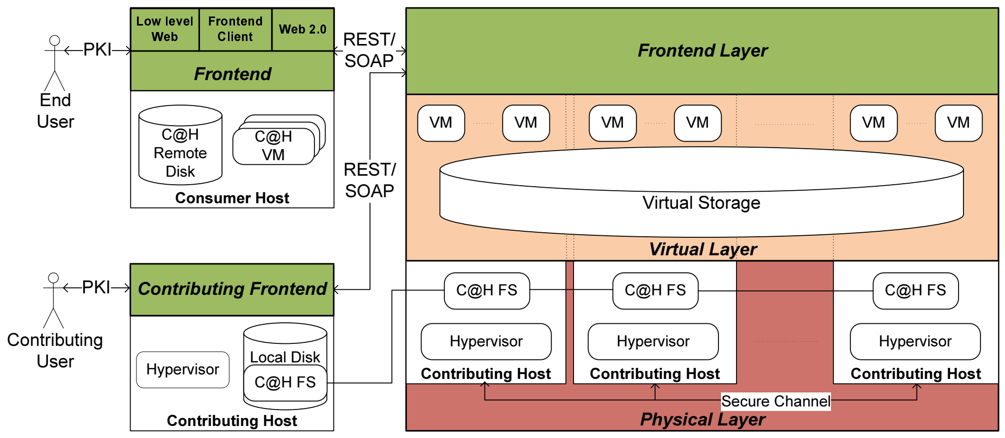

Figure 2. Cloud@Home Separation of Concerns

must be able to enroll, discover, index, assign and reassign, monitor and coordinate resources and services. A problem to face at this level is the compatibility among hardware (architectures, endiannes, etc) and software (operating systems, VM hypervisors, libraries, protocols, applications, etc) and their portability.

- Frontend - abstraction is needed in order to provide users with a high level service oriented point of view of the computing system. The frontend provides a unique, uniform access point to the Cloud. It must allow users to submit functional computing requests only providing requirements and specifications, without any knowledge of the system resources deployment. The system evaluates such requirements and specifications and translates them into physical resources' demand, deploying the elaboration process.
- Security - effective mechanisms are required to provide: identity management, resources and data protection, data confidentiality and integrity.
- Resource and service accessibility, reliability and data consistency - it is necessary to implement redundancy of resources and services, fault tolerance and recovery policies since users voluntarily contribute to the computing, and therefore they can asynchronously, at any time, log out or disconnect from a Cloud.
- Interoperability among Clouds - it should be possible for Clouds to interoperate each other.
- Quality of Service, Service level agreement, and Business models - for selling Cloud computing it is mandatory to provide QoS and SLA management for both commercial and open volunteer Clouds (traditionally best effort), in order to discriminate among the applications to be run.

It is necessary to place and manage such issues into the three layers the functional architecture of Fig. 2. Some of them can be implemented exclusively in one of such layers, others have to be dealt with more than one layer, by considering such issues from the perspective characterized by the layer. In such a way we obtain the layered model of Fig. 3. It implements the core structure of the Cloud@Home server-side middleware, subdivided into *management* and *resource subsystems* operating a further separation of concerns based on implementative characteristics:

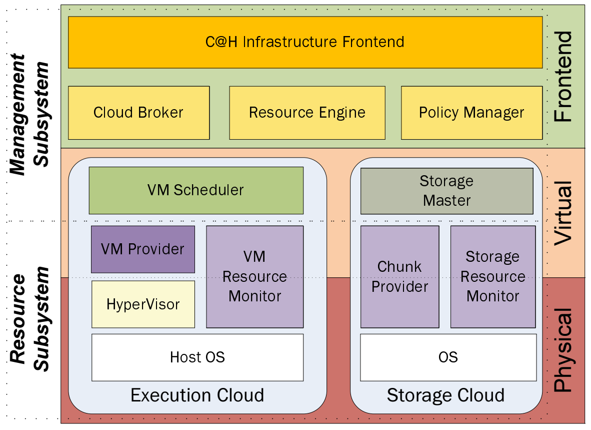

Figure 3. Core Structure of a Cloud@Home Server Middleware.

- *Management subsystem* - is the backbone of the overall system management and coordination composed of six blocks: the *C@H infrastructure frontend*, the *Cloud broker*, the *resource engine*, the *policy manager*, the VM scheduler and the storage master.
- *Resource subsystem* - provides primitives for locally managing the resources (distributed operations), offering different services over the same resources: the *execution Cloud* and the *storage Cloud*.

The two subsystems are strictly interconnected: the management subsystem implements the upper layer of the functional architecture, while the resource subsystem implements the lower level functionalities.

The infrastructure frontend provides tools for Cloud@Home-service provider interactions, forwarding the incoming requests to the lower level blocks. The Cloud broker collects and manages information about the available Clouds and the services they provide (both *functional* and *non-functional* parameters, such as QoS, costs, reliability, *request formats' specifications* for Cloud@Home-foreign Clouds translations, etc). The policy manager provides and implements the Cloud's access facilities. This task falls into the security scope of identification, authentication, permission and identity management.

The resource engine is the hearth of Cloud@Home. It is responsible for the resources' management, the equivalent of a Grid *resource broker* in a broader Cloud environment. To meet this goal, the resource engine applies a hierarchical policy. It operates at higher level, in a centralized way, indexing all the resources of the Cloud. Incoming requests are delegated to VM schedulers or storage masters that, in a distributed fashion, manage the computing or the storage resources respectively, coordinated by the resource engine. In order to manage QoS policies and to perform the resources discovery, the resource engine collaborates with both the Cloud broker and the policy manager.

The resource monitor, both in execution and in storage Clouds, is a functional block split into two parts: the first one runs into the physic host, periodically monitoring its state, the other part is instead implemented by a block of the virtual layer, collecting the information sent by the other part.

The VM provider, the resource monitor and the hypervisor are responsible for managing a VM locally to a physical resource of an execution Cloud. Chunk providers physically store the data into a storage Cloud, that are encrypted in order to achieve the confidentiality goal. This latter in combination with the resource monitor implements the physical layer in a storage Cloud.

IV. The Infrastructure Architecture

Following the (reverse) process of starting from the software/middleware design to implement the Cloud@Home infrastructure, in this section we detail the physical architecture and organization of such infrastructure as the deployment of the middleware core structure of Fig. 3.

In order to implement a distributed architecture, it is necessary to adequately take into account both performance and fault tolerance issues and requirements. Fault tolerance is mainly achieved by redundancy, that however is at the basis of Cloud@Home, and therefore this prefers the distributed approach. Performance issues instead depend on the tasks performed by the blocks: if they cannot be parallelized at all (serial tasks) and/or if they require frequent interactions, in whose great amount of data must be exchanged, the distributed approach can be no more convenient, also from the fault tolerance perspective due to the unreliability of networks. It is therefore necessary a trade-off among the centralized and distributed approach in order to maximize both the fault tolerance and the performance of the Cloud@Home infrastructure and middleware.

So that, in specifying such infrastructure architecture starting from the middleware core structure of Fig. 3, we aggregate functions and blocks that have heavy data dependencies or whose interactions can affect both performance and fault tolerance, and instead disaggregate independent or loosely coupled blocks, deploying these in independent-distributed nodes.

In this way we obtain the architecture of Fig. 4, that pictorially depicts the deployment of the Cloud@Home core structure into the physical infrastructure. Such implementation highlights the hierarchical-distributed approach of Cloud@Home. On top of the hierarchy there are the blocks implementing the management subsystem that, according to the considerations made above, should be deployed into different nodes, one for each block, but however they can also be grouped into the same node. In order to improve fault tolerance, it is necessary to adequately replicate such nodes, in particular if all the management subsystem blocks are deployed into the same unique node.

VM schedulers and storage masters manage smaller or indivisible groups (grid, clusters, multi-core nodes, etc) of resources. They can be designated both globally, by the management subsystem, or locally, by applying self-organizing algorithms such as election mechanisms. A VM scheduler and a storage master can be deployed into the same node, while, obviously, two or more VM schedulers and storage masters cannot coexist in the same physical host. For reliability/availabity/fault tolerance purposes VM schedulers and storage masters should be anyway separated and, moreover, they should be replicated and/or hierarchically organized.

At the bottom of the hierarchy there are the contributing hosts. Each of such leaves contains the blocks, the software for supporting the specific service for what was enrolled into the Cloud. Thus, a node contributing to the execution Cloud has a hypervisor, a VM provider and a VM resource monitor, while a storage Cloud contributing host has its own chunk provider and storage resource monitor. As shown in Fig. 4 and also stated above, it is possible that the same host contributes to both execution and storage Clouds, and therefore it has both execution and storage middleware's components.

V. Conclusions

In this paper we come inside Cloud@Home, a distributed computing paradigm extending and enhancing the Cloud one. Cloud@Home represents a solution for building Clouds, starting from heterogeneous and independent nodes, not specifically conceived for this purpose. This implements a generalization of both Volunteer and Cloud computing by aggregating the computational potentialities of many small,

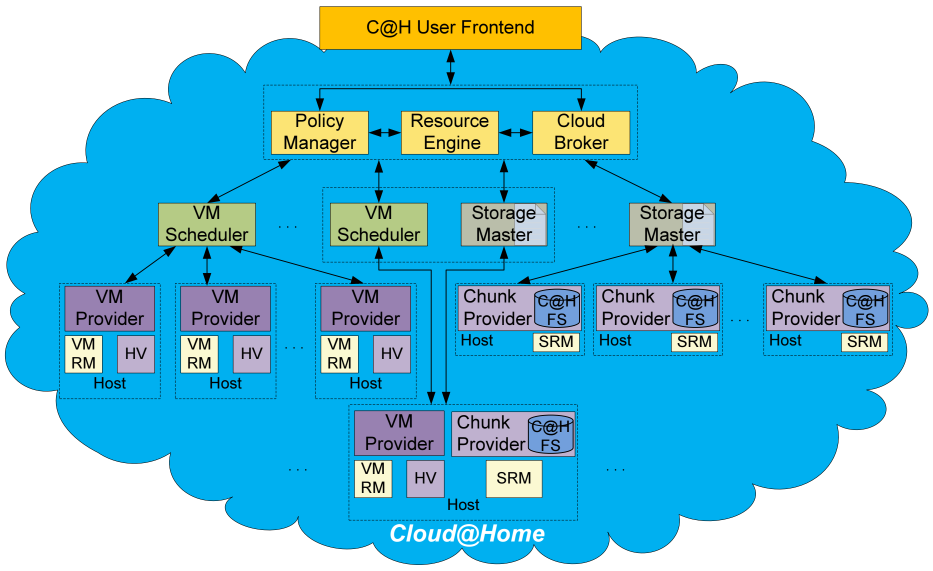

Figure 4. Cloud@home Middleware Infrastructure Deployment.

low power systems, exploiting the "long tail" effect of computing.

Several issues and problems have to be adequately faced for achieving Cloud@Home goals: resources management, security, virtualization, business models, quality of service and service level agreement, intra- and inter-operability, etc. This motivates the necessity of a Cloud@Home middleware specifically conceived for covering them.

In the paper we propose a software architecture implementing such a middleware, and it's deployment into the Cloud@Home infrastructure. In order to obtain this, we firstly identify and logically separate concerns, facets and domains, characterizing a layered model composed of three layers: physical, virtual and frontend. Then, starting from the middleware architecture thus obtained, we try to identify how to physically organize the Cloud@Home infrastructure, by deploying the middleware blocks. In this way we apply a technique widely used in embedded systems contexts, the hardware/software co-design, that we also want to propose as a valid and feasible designing methodology in distributed computing contexts, demonstrating its effectiveness in the Cloud@Home case.

REFERENCES

[1] Amazon Inc. Elastic Compute Cloud [URL]. Amazon, Nov. 2008. http://aws.amazon.com/ec2.

[2] D. P. Anderson and G. Fedak. The computational and storage potential of volunteer computing. In *CCGRID '06*, pages 73–80. IEEE Computer Society, 2006.

[3] F. Balarin, P. Giusto, A. Jurecska, C. Passerone, E. Sentovich, B. Tabbara, M. Chiodo, H. Hsieh, L. Lavagno, A. Sangiovanni-Vincentelli, and K. Suzuki. *Hardware-Software Co-Design of Embedded Systems: The POLIS Approach*, volume 404 of *The Springer International Series in Engineering and Computer Science*. Springer, 1997.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[5] D. Bjorner. *Software Engineering 3: Domains, Requirements, and Software Design*, volume Volume package Software Engineering of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, 2006.

[6] G. Caple. Generic unified systems engineering metamodel: systems design of a single system. In *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE*, volume 1, pages B24–1–9 vol.1, Oct-7 Nov 1998.

[7] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1. W3C, March 2001. http://www.w3.org/TR/wsdl.

[8] M. Co. Azure services platform. http://www.microsoft.com/azure/default.mspx.

[9] V. D. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa. Volunteer computing and desktop cloud: the cloud@home paradigm. In *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications (IEEE NCA09)*. IEEE, 9 - 11 July 2009.

[10] Dell. Dell cloud computing solutions. http://www.dell.com/cloudcomputing.

[11] O. Derin, A. Ferrante, and A. V. Taddeo. Coordinated management of hardware and software self-adaptivity. *Journal of Systems Architecture*, 55(3):170–179, March 2009.

[12] Distributed Systems Architecture Research Group. OpenNEbula Project [URL]. Universidad Complutense de Madrid, 2009. http://www.opennebula.org/.

[13] K. Ebcioglu. The ibm percs project: Hardware-software co-design of a supercomputer for high programmer productivity. Workshop on Application Specific Processors, WASP 2005, 2005.

[14] R. T. Fielding and R. N. Taylor. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, 2002.

[15] I. Foster. Service-oriented science. *Science*, 308(5723), May 2005.

[16] I. Foster and S. Tuecke. Describing the Elephant: The Different Faces of IT as Service. *Queue*, 3(6):26–29, 2005.

[17] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[18] IBM Inc. Blue Cloud project [URL]. IBM, June 2008. http://www-03.ibm.com/press/us/en/pressrelease/22613.wss/.

[19] G. Inc. Google application engine. http://code.google.com/intl/it-IT/appengine/.

[20] E. K. Jackson and J. Sztipanovits. Using separation of concerns for embedded systems design. In *EMSOFT '05: Proceedings of the 5th ACM international conference on Embedded software*, pages 25–34, New York, NY, USA, 2005. ACM.

[21] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov.

[22] RESERVOIR Consortium. RESERVOIR Project [URL], 2009. http://www-03.ibm.com/press/us/en/pressrelease/23448.wss/.

[23] W. W. Staunstrup, Jørgen. *Hardware/Software Co-Design: Principles and Practice*. Springer, 1997.

[24] Sun Microsystem. Network.com [URL]. SUN. http://www.network.com.

[25] Tim O'Reilly. What is WEB 2.0, Sep. 2005. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/ what- is-web-20.html.

[26] University of Chicago-University of Florida-Purdue University-Masaryk University. Nimbus-Stratus-Wispy-Kupa Projects [URL], Jan. 2009. http://workspace.globus.org/clouds/nimbus.html/, http://www.acis.ufl.edu/vws/, http://www.rcac.purdue.edu/ teragrid/resources/#wispy, http://meta.cesnet.cz/cms/opencms/en/docs/clouds.

[27] F. Vahid and T. D. Givargis. *Embedded System Design: A Unified Hardware/Software Introduction*. John Wiley & Sons, Inc, 2002.

[28] VMWare. Understanding Full Virtualization, Paravirtualization, and Hardware Assist, 2007. White Paper.

[29] W3C. Simple object access protocol (soap) 1.2. W3C, April 2007. http://www.w3.org/TR/soap12-part1/.

[30] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl. Scientific Cloud Computing: Early Definition and Experience. In *HPCC '08*, pages 825–830. IEEE Computer Society, 2008.

[31] L.-J. L. Zhang. EIC Editorial: Introduction to the Body of Knowledge Areas of Services Computing. *IEEE Transactions on Services Computing*, 1(2):62–74, April-June 2008.