Chapter 6
Open and Interoperable Clouds: The Cloud@Home Way

Vincenzo D. Cunsolo, Salvatore Distefano, Antonio Puliafito, and Marco Scarpa

Abstract Cloud computing focuses on the idea of *service* as the elementary unit for building any application. Even though Cloud computing was originally developed in commercial applications, the paradigm is quickly and widely spreading in open contexts such as scientific and academic communities. Two main research directions can thus be identified: provide an *open* Cloud infrastructure able to provide and share resources and services to the community; and implement an *interoperable* framework, allowing commercial and open Cloud infrastructures to interact and interoperate. In this chapter, we present the *Cloud@Home* paradigm that proposes to merge *Volunteer* and Cloud computing as an effective and feasible solution for building open and interoperable Clouds. In this new paradigm, users' hosts are not passive interfaces to Cloud services anymore, but can interact (for free or by charge) with other Clouds, which therefore must be able to interoperate.

6.1 Introduction and Motivation

Cloud computing is a distributed computing paradigm that mixes aspects of *Grid computing*, *Internet computing*, *Autonomic computing*, *Utility computing,* and *Green computing*. Cloud computing is derived from the *service-centric perspective* that is quickly and widely spreading in the IT world. From this perspective, all capabilities and resources of a Cloud (usually geographically distributed) are provided to the 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

V.D. Cunsolo (✉), S. Distefano, A. Puliafito, and M. Scarpa
University of Messina, Contrada di Dio, 98166, S. Agata, Messina, Italy
e-mail: vdcunsolo@unime.it; sdistefano@unime.it; apuliafito@unime.it;mscarpa@unime.it

on-demand service provision, *QoS guaranteed offer*, and *autonomous system* for managing hardware, software, and data transparency to the users [25].

In order to achieve such goals, it is necessary to implement a level of abstraction of physical resources, uniforming their interfaces, and providing means for their management, adaptively, to user requirements. This is done through *virtualizations*, *service mashups* (Web 2.0), and *service-oriented architectures* (SOA). These factors make the Kleinrock outlook of computing as the fifth utility [13], following gas, water, electricity, and telephone.

Virtualization [4,23] allows execution of a software version of a hardware machine in a host system in an isolated way. It "homogenizes" resources: problems of compatibility are overcome by providing heterogeneous hosts of a distributed computing environment (the Cloud) using the same virtual machine software.

Web 2.0 [20] provides an interesting way to interface Cloud services, implementing service mashups. It is mainly based on an evolution of JavaScript with improved language constructs (late binding, closures, lambda functions, etc.) and AJAX interactions.

SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains [14]. In SOA, *services* are the mechanism by which needs and capabilities are brought together. SOA defines standard interfaces and protocols that allow developers to encapsulate information tools as services that clients can access without the knowledge of, or control over, their internal workings [8].

An interesting attempt to fix Cloud concepts and ideas is provided in [26] through an ontology that demonstrates a dissection of the Cloud into the five main layers shown in Fig. 6.1. In this, higher layers services can be composed from the services of the underlying layers, which are:

1. *Cloud Application Layer:* provides interface and access-management tools (Web 2.0, authentication, billing, SLA, etc.), specific application services, services mashup tools, etc. to the Cloud end users. This model is referred to as Software as a Service (SaaS).

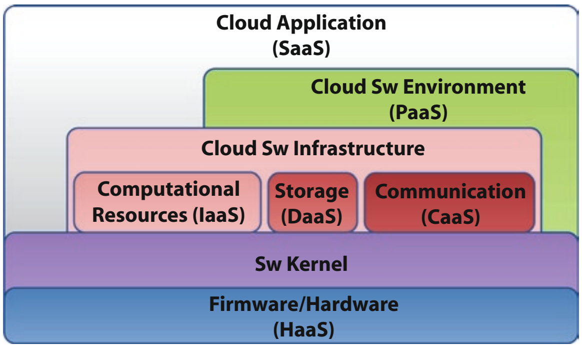

Fig. 6.1 The five main layers of Cloud

2. *Cloud Software Environment Layer*: providers of the Cloud software environments supply the users and Cloud applications' developers with a programming-language-level environment with a set of well-defined APIs. The services provided by this layer are referred to as Platform as a Service (PaaS).
3. *Cloud Software Infrastructure Layer*: provides fundamental resources to other higher-level layers. Services can be categorized into:
   (a) *Computational resources* – provides computational resources (VM) to Cloud end users. Often, such services are dubbed Infrastructure as a Service (IaaS).
   (b) *Data storage* – allows users to store their data at remote disks and access them anytime from any place. These services are commonly known as Data-Storage as a Service (DaaS).
   (c) *Communications* – provides some communication capabilities that are service-oriented, configurable, schedulable, predictable, and reliable. The concept of Communication as a Service (CaaS) emerged toward this goal, to support such requirements.

OAP and REST are examples of interface protocols used with some Cloud computational resources.

4. *Software Kernel*: provides the basic software management for the physical servers that comprise the Cloud. OS kernel, hypervisor, virtual machine monitor, clustering, grid middleware, etc.
5. Hardware and Firmware: form the backbone of the Cloud. End users directly interacting with the Cloud at this layer have huge IT requirements in need of subleasing Hardware as a Service (HaaS).

Great interest in 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, IBM's Blue Cloud, Sun Microsystems Network.com, Microsoft Azure Services Platform, Dell Cloud computing solutions, etc. There are also several scientific activities driving toward Open Cloud-computing middlewares and infrastructures, such as: Reservoir [18], Nimbus-Stratus-Wispy-Kupa [22], OpenNebula [7], Eucalyptus [17], etc. All of them support and provide an on-demand computing paradigm, in the sense that a user submits his/her requests to the Cloud, which remotely, in a distributed fashion, processes them and gives back the results. This client–server model fits the aims and scope of commercial Clouds: the business. But, on the other hand, it represents a restriction for open/scientific Clouds, requiring great amounts of computing-storage resources usually not available from a single open/scientific community. This suggests the necessity to collect such resources from different providers and/or contributors who could share their resources with the specific community, perhaps by making "symbiotic" federations. In fact, one of the most successful paradigms in such contexts is *Volunteer computing*.

Volunteer computing (also called *Peer-to-Peer computing*, *Global computing,* or *Public computing*) uses computers volunteered by their owners as a source of computing power and storage to provide distributed scientific computing [2].

It is the basis of the *"@home"* philosophy of sharing/donating network connected resources for supporting distributed scientific computing.

We believe that the Cloud-computing paradigm is also applicable at lower scales, from the single contributing user who shares his/her desktop, to research groups, public administrations, social communities, and small and medium enterprises, who can make their distributed computing resources available to the Cloud. Both free sharing and pay-per-use models can be easily adopted in such scenarios.

From the utility point of view, the rise of the "techno-utility complex" and the corresponding increase in computing resource demands, in some cases growing dramatically faster than Moore's Law, predicted by the Sun CTO Greg Papadopoulos in the *red shift theory* for IT [15], could take us in a close future, toward an *oligarchy*, a lobby or a trust of few big companies controlling the whole computing resources market.

To avoid such a pessimistic but achievable scenario, we suggest addressing the problem in a different way: instead of building costly private *data centers* that the Google CEO, Eric Schmidt, likes to compare with the prohibitively expensive cyclotrons [3], we propose a more "democratic" form of Cloud computing, in which the computing resources of single users accessing the Cloud can be shared with others in order to contribute to the elaboration of complex problems.

As this paradigm is very similar to the Volunteer computing one, it can be named as *Cloud@Home*. Both hardware and software compatibility limitations and restrictions of Volunteer computing can be solved in Cloud computing environments, allowing to share both hardware and software resources and/or *services*.

The Cloud@Home paradigm could also be applied to commercial Clouds, establishing an *open computing-utility market* where users can both buy and sell their services. Since the computing power can be described by a "long-tailed" distribution, in which a high-amplitude population (Cloud providers and commercial data centers) is followed by a low-amplitude population (small data centers and private users) that gradually "tails off" asymptotically, Cloud@Home can catch the *Long Tail* effect [1], providing similar or higher computing capabilities than commercial providers' data centers, by grouping small computing resources from many single contributors.

In the following, we demonstrate how it is possible to realize all these aims through the Cloud@Home paradigm. In Section 2, we describe the functional architecture of the Cloud@Home infrastructure, and in Section 3, we characterize the blocks implementing the functions previously identified into the Cloud@Home core structure. Section 4 concludes the chapter by recapitulating our work and discussing challenges and future work.

6.2 Cloud@Home Overview

The idea behind Cloud@Home is to reuse *"domestic"* computing resources to build voluntary contributors' Clouds that are interoperable and, moreover, interoperate with other foreign, and also commercial, Cloud infrastructures. With Cloud@Home, anyone can experience the power of Cloud computing, both actively by providing his/her own resources and services, and passively by submitting his/her applications and requirements.

6.2.1 Issues, Challenges, and Open Problems

Ian Foster summarizes the computing paradigm of the future as follows [9]: "... we will need to support on-demand provisioning and configuration of integrated 'virtual systems' providing the precise capabilities needed by an end user. We will need to define protocols that allow users and service providers to discover and hand off demands to other providers, to monitor and manage their reservations, and arrange payment. We will need tools for managing both the underlying resources and the resulting distributed computations. We will need the centralized scale of today's Cloud utilities, and the distribution and interoperability of today's Grid facilities...."

We share all these requirements, but in a slightly different way: we want to actively involve users into such a new form of computing, allowing them to create their own interoperable Clouds. In other words, we believe that it is possible to export, apply, and adapt the *"@home"* philosophy to the Cloud-computing paradigm. In this way, by merging Volunteer and Cloud computing, a new paradigm can be created: *Cloud@Home*. This new computing paradigm gives back the power and control to users, who can decide how to manage their resources/services in a global, geographically distributed context. They can voluntarily sustain scientific projects by freely placing their resources/services at the scientific research centers' disposal, or can earn money by selling their resources to Cloud-computing providers in a pay-per-use/share context.

Therefore, in Cloud@Home, both the commercial/business and volunteer/scientific 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/or sell their resources/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 in to the Grid and Volunteer philosophy (*best effort*) the concept of quality.

Cloud@Home can also be considered as a generalization and a maturation of the @home philosophy: a context in which users voluntarily share their resources without compatibility problems. This allows knocking down both hardware (processor bits, endianness, architecture, and network) and software (operating systems, libraries, compilers, applications, and middlewares) barriers of Grid and Volunteer computing. Moreover, Cloud@Home allows users to share not only physical resources, as in @home projects or Grid environments, but any kind of service. The *flexibility* and *extensibility* of Cloud@Home can allow to easily arrange, manage, and make available with significant computing resources (greater than those in Clouds, Grids, and/or @home environments) to everyone who owns a computer. Another significant improvement of Cloud@Home with regard to Volunteer computing paradigms is the QoS/SLA management: starting from the credit management system and other similar experiments on QoS, a mechanism for adequately monitoring, ensuring, negotiating, accounting, billing, and managing, in general, QoS and SLA will be implemented.

On the other hand, Cloud@Home can be considered as the enhancement of the Grid-Utility vision of Cloud computing. In this new paradigm, user's hosts are not passive interfaces to Cloud services, but can be actively involved in computing. At worst, single nodes and services could be enrolled by the Cloud@Home middleware to build own-private Cloud infrastructures that can with interact with other Clouds.

The Cloud@Home motto is: *heterogeneous hardware for homogeneous Clouds*. Thus, the scenario we prefigure is composed of several coexisting and interoperable Clouds, as depicted in Fig. 6.2. *Open Clouds* (yellow) identify open VO operating for free Volunteer computing; *Commercial Clouds* (blue) characterize entities or companies selling their computing resources for business; and *Hybrid Clouds* (green) can both sell or give for free their services. Both Open and Hybrid Clouds can interoperate with any other Clouds, as well as Commercial, while these latter can interoperate if and only if the Commercial Clouds are mutually *recognized*. In this way, it is possible to make *federations* of heterogeneous Clouds that can work together on the same project. Such a scenario has to be implemented transparently for users who do not want to know whether their applications are running in homogeneous or heterogeneous Clouds. The differences among homogeneous and heterogeneous Clouds are only concerned with implementation issues, mainly affecting the resource management: in the former case, resources are managed locally to the Cloud; in heterogeneous Clouds, interoperable services have to be implemented in order to support discovery, connectivity, translation, and negotiation requirements amongst Clouds.

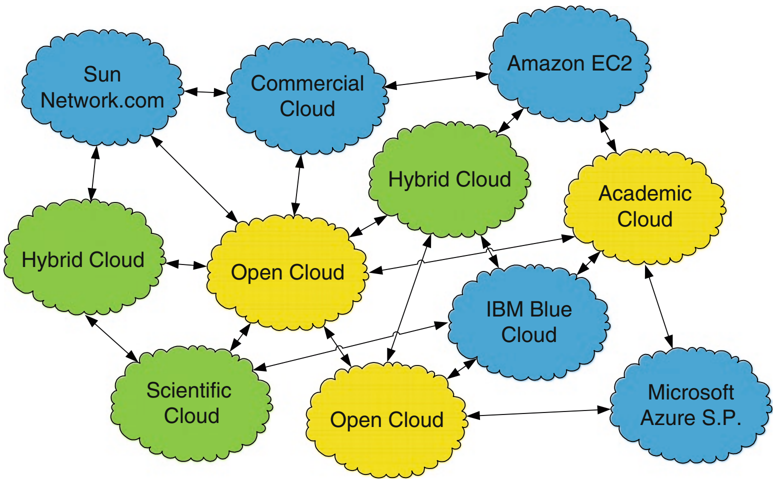

Fig. 6.2 Co-existing and interoperable Clouds anticipated for the Cloud@Home Scenario

The overall infrastructure must deal with the high dynamism of its nodes/resources, allowing to move and reallocate data, tasks, and jobs. It is therefore necessary to implement a lightweight middleware, specifically designed to optimize migrations. The choice of developing such middleware on existing technologies (as done in Nimbus-Stratus starting from Globus) could be limiting, inefficient, or not adequate from this point of view. This represents another significant enhancement of Cloud@Home against Grid: a lightweight middleware allows to involve limited resources' devices into the Cloud, mainly as consumer hosts accessing the Cloud through "thin client" but also, in some specific applications, as contributing hosts implementing (light) services according to their availabilities. Moreover, the Cloud@Home middleware does not influence code writing as Grid and Volunteer computing paradigms do.

Another important goal of Cloud@Home is *security*. Volunteer computing has security concerns, while the Grid paradigm implements complex security mechanisms. Virtualization in Clouds implements isolation of services, but does not provide any protection from local access. With regard to security, the specific goal of Cloud@Home is to extend the security mechanisms of Clouds to the protection of data from local access. As Cloud@Home is composed of an amount of resources potentially larger than commercial or proprietary Cloud solutions, its *reliability* can be compared with Grid or the Volunteer computing and should be greater than other Clouds.

Lastly, *interoperability* is one of the most important goals of Cloud@Home. This is an open problem in Grid, Volunteer, and Cloud computing, which we want to address in Cloud@Home.

The most important issues that should be taken into account in order to implement such a form of computing can be listed as follows:

- *Resources and Services management* – a mechanism for managing resources and services offered by Clouds is mandatory. This 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 resources and services 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 resource demands, deploying the elaboration process. Another aspect concerning the frontend is the capability of customizing Cloud services and applications.
- *Security* – effective mechanisms are required to provide authentication, 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, and hosts' recovery policies because users voluntarily contribute to the computing, and therefore, can asynchronously, at any time, log out or disconnect from the Cloud.

- *Interoperability among Clouds* – it should be possible for Clouds to interoperate.
- *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) to discriminate among the applications to be run.

6.2.2 Basic Architecture

A possible Cloud@Home architecture that could address the issues listed earlier is shown in Fig. 6.3, which has been adapted to the ontology provided in [26] and reported in Fig. 6.1. Two types of users are distinguished in such an architecture according to the role that they assume in the Cloud: *end users*, if they only interface the Cloud for submitting requests, and/or *contributing users* if they make available their resources and services for building up and supporting the Cloud. According to this point of view, the Cloud is composed of several *contributing hosts* offered by the corresponding contributing users to end users who interact with the Cloud and submit their requests through their *consumer hosts*. To access a Cloud, both contributing and end users must authenticate themselves into the system. One of the main enhancements of Cloud@Home is that a user/host can be contributing and/or end user/consumer host, establishing a symbiotic mutual interaction with the Cloud.

Such an architecture will be described below by identifying and detailing tasks and functions of each of the five layers characterized in the Cloud ontology presented in Section 1.

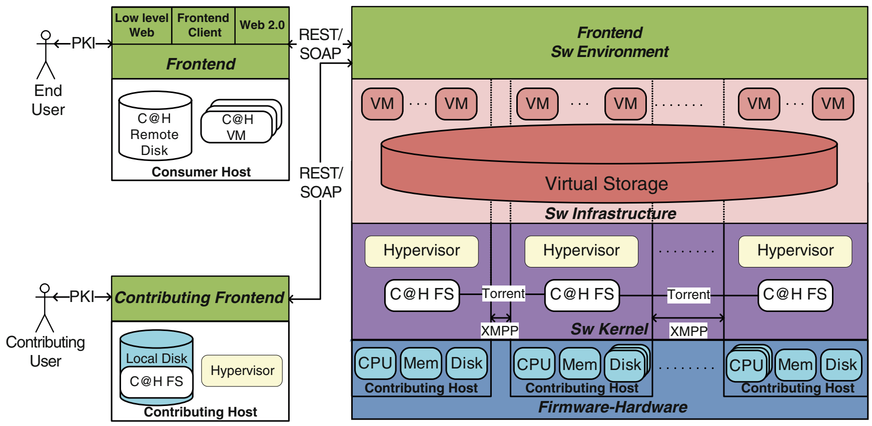

Fig. 6.3 Basic architecture of Cloud@Home

6.2.2.1 Software Environment

The Cloud@Home software environment implements the user-infrastructure frontend interface. It is responsible for the resources and services management (enrolling, discovery, allocation, coordination, monitoring, scheduling, etc.) from the global Cloud system's perspective. It also provides tools, libraries, and APIs for translating user requirements into physical resource demands. Moreover, in commercial Clouds, it must be able to negotiate the QoS policy to be applied (SLA), thus monitoring for its fulfillment and, in case of unsatisfactory results, adapting the computing workflow to such QoS requirements.

If the Cloud's available resources and services do not satisfy the requirements, the frontend provides mechanisms for requesting further resources and services from other Clouds, both open and/or commercial. In other words, the Cloud@Home frontend implements the interoperability among Clouds, also checking for service reliability and availability. To improve reliability and availability of services and resources, especially if QoS policies and constraints have been specified, it is necessary to replicate services and resources by introducing redundancy.

The Cloud@Home software environment is split into two parts, as shown in Fig. 6.3: the server side, implementing resource management and related problems, and the client side, providing mechanisms and tools for authenticating, enrolling, accessing, and interacting with the Cloud services and resources. The client frontend is distinguished according to the role assumed by the user/host: for end users, only a *thin* client able to interact with the frontend server and to submit requests to the Cloud must be installed into the consumer hosts; for contributing users, contributing hosts must provide the software for interfacing with the Cloud@Home frontend server and for supporting the Cloud (C@H FS library, storage space, and/or hypervisor according to the service supported).

In a widely distributed system that is globally spread, the knowledge of resource accesses and uses assumes great importance. To access and/or use the Cloud services, a generic user first authenticates him/herself and then specifies whether he/she wants to make available his/her resources and services for sharing, or he/she only uses the Cloud resources for computing. The frontend provides means, tools, and policies for managing users. The best mechanism to achieve secure authentication is the *Public Key Infrastructure* (PKI) [21], better if combined with smartcard devices that, through a trusted certification authority, ensure user identification. In order to avoid multiple authentications, a mechanism of authentication management and credential delegation, such as *single sign-on* (SSO), must be provided by the server frontend.

Referring to Fig. 6.3, three alternative solutions can be offered to end users by the software environment for accessing a Cloud: (a) Cloud@Home frontend client, (b) Web 2.0 user interface, and (c) low-level Web interface (directly specifying REST or SOAP queries). These also provide mechanisms for customizing user applications by composing (service mashup and SOA) and submitting own services.

6.2.2.2 Software Infrastructure

The virtualization of physical resources offers a homogeneous view of Cloud's services and resources to end users. Two basic services are provided by the software infrastructure to the software environment, and consequently, to end users: *execution* and *storage* services.

The execution service, implementing the computational resources sublayer of Fig. 6.1, allows to create and manage virtual machines. A user, sharing his/her resources within a Cloud@Home, allows the other users of the Cloud to execute and manage virtual machines locally at his/her node, according to policies and constraints negotiated and monitored through the software environment. In this way, a Cloud of virtual machine's executors is established, where virtual machines can migrate or can be replicated in order to achieve reliability, availability, and QoS targets. As shown in Fig. 6.3, from the end user's point of view, an execution Cloud is seen as a set of virtual machines available and ready-to-use. The virtual machines' *isolation* implements protection and therefore security. This security is ensured by the hypervisor that runs the virtual machine's code in an isolated scope, similarly to a sandbox, without affecting the local host environment.

The storage service implements a storage system distributed across the storage hardware resources composing the Cloud, highly independent of them because data and files are replicated according to QoS policies and requirements to be satisfied. From the end user's point of view, a storage Cloud appears as a locally mounted remote disk, similar to a Network File System or a Network Storage. Tools, libraries, and API for interfacing to storage Clouds are provided by the frontend client to end users, while the service is implemented by the Cloud@Home software infrastructure and software kernel.

In a distributed environment where any user can host a part of private data, it is necessary to protect such data from unauthorized accesses (data security). A way to obtain data *confidentiality* and *integrity* could be cryptography, as better explained in the software kernel description.

6.2.2.3 Software Kernel

The software kernel provides infrastructure, mechanisms, and tools to the software for locally managing the physical resources of the Cloud in order to implement execution and storage services.

Cloud@Home negotiates with users who want to join a Cloud about his/her contribution. This mechanism involves the software kernel that provides tools for reserving execution and/or storage resources for the Cloud, and monitors these resources so that constraints, requirements, and policies specified are not violated. This ensures reliability and availability of the resources, avoiding overloading of the local system and therefore reducing the risk of crashes.

To implement the execution service in a generic device or to enroll it into an execution Cloud, the device must have a hypervisor ready to allocate and run virtual machines, as shown in Fig. 6.3 . If a storage service is installed into the device, a portion of the local storage system must be dedicated for hosting the Cloud data. In such cases, the Cloud@Home file system is installed into the devices' shared storage space.

The software kernel also implements data security (integrity and confidentiality), ensuring that stored data cannot be accessed by those who physically host them (*insider attacks, identity thefts, account hijacking*, etc.). We propose an approach that combines the inviolability of the Public Key Infrastructure asymmetric cryptography and the speed of symmetric cryptography (details in [6]). Data are first encrypted by the symmetric key and then stored into the selected host with the symmetric key encrypted by the user private key. This ensures that only authorized users can decrypt the symmetric key and consequently access the data.

In order to implement secure and reliable connections amongst nodes, we choose the *Extensible Messaging and Presence Protocol* (XMPP) protocol [19]. XMPP is an open technology for real-time communication, which powers a wide range of applications including instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data, also supporting security features. However, as the data stored in a Cloud@Home storage are encrypted, it is not necessary to use a secure channel for data transfers, and hence, a more performant protocol, such as BitTorrent [5] can be used. The XMPP secure channel is required for sending and receiving nonencrypted messages and data to/from remote hosts.

6.2.2.4 Firmware/Hardware

The Cloud@Home firmware/hardware layer is composed of a *"cloud"* of generic contributing nodes and/or devices geographically distributed across the Internet. They provide the physical-hardware resources to the upper layers for implementing the execution and storage services.

6.2.3 Application Scenarios

Several possible application scenarios can be imagined for Cloud@Home:

- *Research centers, public administrations, and communities* – the Volunteer computing inspiration of Cloud@Home provides means for the creation of open, interoperable Clouds for supporting scientific purposes, overcoming the portability and compatibility problems highlighted by the @home projects. Similar benefits could be experienced in public administrations and open communities (social networks, peer-to-peer, gaming, etc). Through Cloud@Home, it could be possible to implement resources and service management policies with QoS requirements (characterizing the scientific project importance) and specifications (QoS classification of resources and services available). A new deal for Volunteer computing, since this latter does not take into consideration QoS, follows a best effort approach.

- *Enterprises* – planting a Cloud@Home computing infrastructure in business-commercial environments can bring considerable benefits, especially in small and medium, as well as big enterprises. Usually, in every enterprise, there exists a capital of stand-alone computing resources dedicated to a specific task (office automation, monitoring, designing, and so on). Since such resources are only (partially) used in office hours, through Internet connectivity, it becomes possible to build up a Cloud@Home data center, in which shared services are allocated (web server, file server, archive, database, etc.) without any compatibility constraints or problems.
- The interoperability amongst Clouds allows to buy computing resources from commercial Cloud providers if needed or, otherwise, to sell the local Cloud computing resources to the same or different providers. This allows reducing and optimizing business costs according to QoS/SLA policies, improving performances and reliability. For example, this paradigm allows dealing with the *peaks economy*: data centers could be sized for managing the medium case, and worst cases (peaks) could be managed by buying computing resources from Cloud providers. Moreover, Cloud@Home drives towards resource rationalization: all the business processes can be securely managed over the web, allocating resources and services where needed. In particular, this can improve marketing and trading (E-commerce), making available a lot of customizable services to sellers and customers. The interoperability could also point to another scenario, in which private companies buy computing resources in order to resell them (*subcontractors*).
- *Ad-hoc networks, wireless sensor networks, and home automation* – the Cloud-computing approach, in which both software and computing resources are owned and managed by service providers, eases the programmers' efforts in facing device heterogeneity problems. Mobile application designers should start to consider that their applications, besides needing to be usable on a small device, will need to interact with the Cloud. Service discovery, brokering, and reliability are important issues, and services are usually designed to interoperate. In order to consider the arising consequences related to the access of mobile users to service-oriented grid architecture, researchers have proposed new concepts such as mobile dynamic virtual organizations [24].
- An open research issue is whether or not a mobile device should be considered as a service provider of the Cloud itself. The use of modern mobile terminals, such as smart-phones, not just as Web service requestor but also as mobile hosts that can themselves offer services in a true mobile peer-to-peer setting, is also discussed in [16]. Context-aware operations involving control and monitoring, data sharing, synchronization, etc, could be implemented and exposed as Cloud@Home Web services. Cloud@Home could be a way to implement *Ubiquitous* and *Pervasive* computing: many computational devices and systems can be engaged simultaneously for performing ordinary activities, and may not necessarily be aware of the fact that they are doing so.

6.3 Cloud@Home Core Structure

Once the functional architecture of Cloud@Home has been introduced, it is necessary to characterize the blocks implementing the functions thus identified. These blocks are pictorially depicted in the layered model of Fig. 6.4 that reports the core structure of the overall system implementing the Cloud@Home server-side. As done for the functional architecture, the core structure is also specified by following the Cloud ontology characterized in Fig. 6.1. Moreover, the Cloud@Home core structure is subdivided into two subsystems: *management* and *resource subsystems*. Such subsystems are strictly interconnected: the management subsystem implements the upper layer of the functional architecture, while the resource subsystem implements the lower level functionalities.

Figure 6.5 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 can be deployed into different servers/nodes, one for each block, or can be grouped into the same node. Nevertheless, in order to achieve reliability and availability goals, it is necessary to adequately

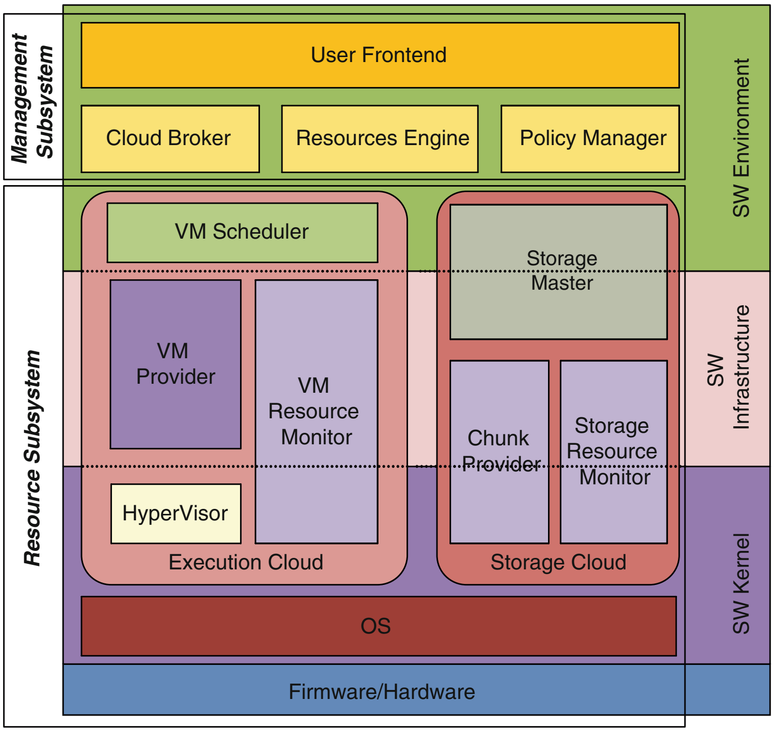

Fig. 6.4 Cloud@Home Core Structure Organisation

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 groups (grid, clusters, multi-core nodes, etc.) of resources. They can be designated both globally by the management subsystem and/or locally by applying self-organizing/autonomic algorithms such as election mechanisms. A VM scheduler and a storage master can be deployed into the same node/server, while, obviously, two or more VM schedulers/storage masters cannot coexist in the same node. For reliability/availability purpose, they can also be replicated and/or hierarchically organized.

At the bottom of the hierarchy, there are the contributing hosts. Each contains 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 a chunk provider and a storage resource monitor. As shown in Fig. 6.4 and also stated earlier, it is possible that the same host contributes to both execution and storage Clouds, and therefore, has both execution and storage components.

6.3.1 Management Subsystem

In order to enroll and manage the distributed resources and services of a Cloud, providing a unique point of access for them, it is necessary to adopt a centralized approach that is implemented by the management subsystem. It is composed of four parts: the *user frontend*, the *Cloud broker*, the *resource engine,* and the *policy manager*.

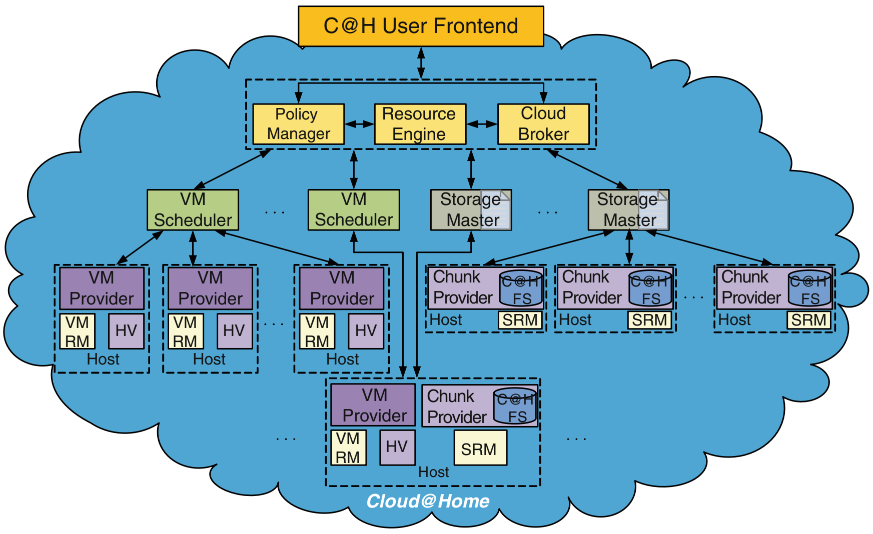

Fig. 6.5 Cloud@Home Core Structure Infrastructure Deployment.

The user frontend provides tools for Cloud@Home-user interactions. It collects and manages the users' requests issued by the Cloud@Home clients. All such requests are transferred to the blocks composing the underlying layer (resource engine, Cloud broker, and policy manager) for processing.

An important task carried out by the user frontend is the Clouds interoperability, implemented point-to-point, connecting the interface of the Clouds wishing to interoperate. If one of the Clouds does not have the Cloud@Home core structure of Fig. 6.3, it is necessary to translate the requests between Cloud@Home and foreign Clouds formats, a task delegated by the user frontend to the Cloud broker. 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, and reliability, *request formats' specifications* for Cloud@Home-foreign Cloud translations, etc.).

The policy manager provides and implements the Cloud's access facilities. This task falls into the security scope of identification, authentication, authorization, and permissions management. To achieve this target, the policy manager uses an infrastructure based on PKI, smartcard devices, Certification Authority, and SSO. The policy manager also manages the information about users' QoS policies and requirements.

The resource engine is the heart 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 a 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 storage resources, respectively, coordinated by the resource engine.

The management subsystem is implemented as a centralized subsystem managing the whole infrastructure. Although this solution introduces a single point of failure into the architecture, this is the only possible way to manage resource QoS, SLA, dynamic provisioning, and monitoring because there has to be a subsystem that aggregates information and has to know the condition of the whole infrastructure, there needs to be a coordinator. Reliability, availability, and fault-tolerance issues can be achieved by replicating the management subsystem and its components, adequately managing the consistency of redundant replicas.

6.3.2 Resource Subsystem

The resource subsystem contains all the blocks implementing the local and distributed management functionalities of Cloud@Home. This subsystem can be logically split into two parts offering different software infrastructure services: the *execution Cloud* and the *storage Cloud*. The management subsystem is also able to merge them, providing a unique Cloud that can offer both execution and/or storage services.

The execution Cloud provides tools for managing virtual machines according to users' requests and requirements coming from the management subsystem. It is composed of four blocks: *VM scheduler*, *VM provider*, *VM resource monitor,* and *hypervisor*.

The VM Scheduler is a peripheral resource broker of the Cloud@Home infrastructure, to which the resource engine delegates the management of computing/execution resources and services of the Cloud. It establishes which, what, where, and when to allocate a VM; moreover, it is responsible for moving and managing VM services. From the end user's point of view, a VM is allocated somewhere on the Cloud; therefore, its migration is transparent for the end user that is not aware of any VM migration mechanism. However, some problems can affect VM migrations into the Cloud@Home environment. As the nodes implementing the Cloud are, generally, widely distributed across the Internet, for migrating a VM with its entire context from one node to another (remote) node, great transfer delays are introduced. In a highly dynamic environment, where VM migrations could be highly frequent, this could become a serious problem.

A possible solution to such a problem is the introduction of technique-based difference algorithms, similar to the one implemented in the *union file system* (UnionFS) [27]. In each node contributing to the execution Cloud of the Cloud@Home infrastructure, redundant basic VM images must be available (if possible). Thus, in case of migration, starting from the selected data/files comparison algorithm (diff), instead of transferring the whole VM with its context, a lighter (diff) file only containing the differences between a new VM and the one to migrate is sent to the destination host, which recomposes the original VM starting from a new VM instance and runs it. This technique can considerably reduce the amount of data to transfer, and consequently the corresponding transfer times.

Differentiation techniques might be appropriate for moving VM disk images (although they would require some fundamental restrictions to be placed on the images that could be used), but they do not address the problem of migrating VM memory state. Such a problem could be addressed by exploiting specific and advanced live migration techniques implementing reduced bandwidth usage, just-in-time live migration behavior, and live migration across WAN, mainly based on compression algorithms [11,12].

The VM provider, the VM resource monitor, and the hypervisor are responsible for managing a VM locally to a physical resource. A VM provider exports functions for allocating, managing, migrating, and destroying a virtual machine on the corresponding host. The VM resource monitor allows taking the local computing resources under control, according to requirements and constraints negotiated in the setup phase with the contributing user. If during a virtual machine execution, the local resources crash or become insufficient to keep the virtual machine running, the VM resource monitor asks the scheduler to migrate the VM elsewhere.

In order to implement the storage Cloud, we specify the *Cloud@Home file system* (FS), adopting an approach similar to the Google FS [10]. The Cloud@Home FS splits data and files into *chunks* of fixed or variable size, depending on the storage resources available. The architecture of the storage file system is hierarchical: data chunks are physically stored on *chunk providers* and corresponding *storage masters* index the chunks through specific *file indexes* (FI). The storage master is the directory server, indexing the data stored in the associated chunk providers. It directly interfaces with the resource engine to discover the resources storing data. In this context, the resource engine can be considered, in its turn, as the directory server indexing all the storage masters. To improve the storage Cloud reliability, storage masters must be replicated. Moreover, a chunk provider can be associated to more than one storage master.

In order to avoid a storage master becoming a bottleneck, once the chunk providers have been located, data transfers are implemented by directly connecting end users and chunk providers. Similar techniques to the ones discussed about VM schedulers can be applied to storage masters for improving performance and reliability of the storage Clouds.

Chunk providers physically store the data that, as introduced earlier, are encrypted to achieve the confidentiality goal. Data reliability can be improved by replicating data chunks and chunk providers, consequently updating the corresponding storage masters. In this way, a corrupted data chunk can be automatically recovered and restored through the storage masters, without involving the end user.

In order to achieve QoS/SLA requirements in a storage Cloud, it is necessary to periodically monitor its storage resources, as done in the execution Cloud for VM. For this reason, in the Cloud@Home core structure of Fig. 6.3, we have introduced a specific storage resource monitor block. As it monitors the state of a chunk provider, it is physically located and deployed into each chunk provider composing the storage Cloud. The choice of replicating the resource monitor in both execution and storage Clouds is motivated by the fact that we want to implement two different, separated, and independent services.

6.4 Conclusions

In this chapter, we have discussed an innovative computing paradigm merging volunteer contributing and Cloud approaches into Cloud@Home. This proposal represents a solution for building Clouds, starting from heterogeneous and independent nodes, not specifically conceived for this purpose. Cloud@Home implements a generalization of both Volunteer and Cloud computing by aggregating the computational potentialities of many small, low-power systems, exploiting the long-tail effect of computing.

In this way, Cloud@Home opens the Cloud computing world to scientific and academic research centers, as well as to public administration and communities, and potentially single users: anyone can voluntarily support projects by sharing his/her resources. On the other hand, it opens the utility computing market to the single user who wants to sell his/her computing resources. To realize this broader vision, several issues must be adequately taken into account: reliability, security, portability of resources and services, interoperability among Clouds, QoS/SLA, and business models and policies.

It is necessary to have a common understanding regarding an ontology that fixes concepts, such as resources, services, virtualization, protocol, format, interface, and corresponding metrics, including Clouds' functional and nonfunctional parameters (QoS, SLA, and so on), which must be translated into specific interoperable standards.

References

1. Anderson C. (2006) The long tail: how endless choice is creating unlimited demand. Random House Business Books, London
2. Anderson DP, Fedak G (2006) The computational and storage potential of volunteer computing. In: CCGRID '06, pp 73–80
3. Baker S (2008, December 24) Google and the wisdom of clouds. BusinessWeek. http://www.businessweek.com/magazine/content/07_52/b4064048925836.htm
4. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A (2003) Xen and the art of virtualization. In: Proceedings of the nineteenth ACM symposium on operating systems principles (SOSP '03), ACM, pp 164–177
5. Cohen B (2008) The BitTorrent protocol specification. BitTorrent.org. http://www.bittorrent.org/beps/bep_0003.html
6. Cunsolo VD, Distefano S, Puliafito A, Scarpa M (2009) Implementing data security in grid environment. Enabling technologies, IEEE international workshops on **0**, pp 177–182
7. Distributed Systems Architecture Research Group: OpenNEbula Project [URL]. Universidad Complutense de Madrid (2009). http://www.opennebula.org/
8. Foster I (2005) Service-oriented science. Science 308(5723)
9. Foster I (2008) There's grid in them thar clouds. Ian Foster's blog. http://ianfoster.typepad.com/blog/2008/01/theres-grid-in.html
10. Ghemawat S, Gobioff H, Leung ST (2003) The google file system. SIGOPS Oper Syst Rev 37(5):29–43
11. Hacking S, Hudzia B (2009) Improving the live migration process of large enterprise applications. In: Proceedings of the 3rd international workshop on virtualization technologies in distributed computing (VTDC '09), ACM, pp 51–58
12. Jin H, Deng L, Wu S, Shi X, Pan X (2009) Live virtual machine migration with adaptive, memory compression. In: IEEE cluster computing and workshops (CLUSTER '09), pp 1–10
13. Kleinrock L (2005) A vision for the internet. ST J Res 2(1):4–5
14. MacKenzie CM, Laskey K, McCabe F, Brown PF, Metz R, Hamilton BA (2006) Reference model for service oriented architecture 1.0. http://docs.oasis-open.org/soa-rm/v1.0/
15. Martin R (2007, August 20) The red shift theory. InformationWeek. http://www.informationweek.com/news/hardware/showArticle.jhtml?articleID=201800873
16. Narayana SS, Jarke M, Prinz W (2006) Mobile web service provisioning. In: AICT-ICIW '06, IEEE Computer Society, p 120
17. Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D (2008) The eucalyptus open-source cloud-computing system. In: Proceedings of cloud computing and its applications (2008)
18. Reservoir Consortium: Reservoir Project [URL] (2009). http://www-03.ibm.com/press/us/en/pressrelease/23448.wss/
19. Saint-Andre P, Tronçon R, Smith K (2008) XMPP: the definitive guide: building real-time applications with jabber technologies, rough cuts version edn. O'Reilly
20. Tim O'Reilly: What is WEB 2.0 (2005). http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

21. Tuecke S, Welch V, Engert D, Pearlman L, Thompson M (2004) Internet X.509 Public Key Infrastructure (PKI) proxy certificate profile
22. University of Chicago-University of Florida-Purdue University-Masaryk University: Nimbus-Stratus-Wispy-Kupa Projects [URL] (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
23. VMWare: Understanding Full Virtualization, Paravirtualization, and Hardware Assist (2007). White Paper
24. Waldburger M, Stiller B (2006) Toward the mobile grid: service provisioning in a mobile dynamic virtual organization. In: IEEE international conference on computer system and application, pp 579–583
25. Wang L, Tao J, Kunze M, Castellanos AC, Kramer D, Karl W (2008) Scientific cloud computing: early definition and experience. In: HPCC ’08, pp 825–830
26. Youseff L, Butrico M, Da Silva D (2008) Toward a unified ontology of cloud computing. In: Grid computing environments workshop (GCE ’08), pp 1–10
27. Zadok E, Iyer R, Joukov N, Sivathanu G, Wright CP (2006)) On incremental file system development. ACM Trans Storage 2((2):161–196