The Personal Cloud — Design, Architecture and Matchmaking Algorithms for Resource Management

Adiseshu Hari
Bell Labs, USA
Adiseshu.Hari@alcatel-lucent.com

Ramesh Viswanathan
Bell Labs, USA
Ramesh.Viswanathan@alcatel-lucent.com

T.V. Lakshman
Bell Lab, USA
T.v.Lakshman@alcatel-lucent.com

Y. J. Chang
ITRI, Taiwan
Yuh-Jye.Chang@itri.org.tw

Abstract

We introduce the notion of a Personal Cloud — a collection of Virtual Machines (VMs) running on unused computers at the edge. The Personal Cloud provides an ideal solution for the secure sharing of compute and storage resources across peers in a resource and application agnostic manner, and facilitates new computational paradigms such as datacenter-less, distributed virtual clouds. We provide and implement solutions for the challenges of managing a Personal Cloud, such as IP address sharing, bandwidth sharing and isolation from local home network traffic. We also propose and implement a provably optimal solution to the resource management problem, allowing peers to share VMs across their individual Personal Clouds by specifying their resource offers and requests, and verify its performance via detailed simulations.

1 Introduction

Each endpoint of the Internet represents resources such as CPU, storage and bandwidth which are the exclusive property of the owner of the endpoint. There are benefits, however, in sharing unused resources, for example, two peers could agree to do a mutual backup, or a group could get together to create a shared repository by using unused computers. A fundamental challenge in resource sharing, however, is the security and privacy needed for hosting remote jobs on local resources. In this paper, we describe an architecture which we call the *Personal Cloud*, which packages unused edge resources into a standard Infrastructure as a Service (IaaS) Cloud such that arbitrary VMs and applications can be safely run locally and accessed remotely.

Since the birth of the Internet, there have been numerous attempts to create platforms for remote and distributed computation. Today, we have Grid computing which seeks to harness unused enterprise resources, Planetlab [1] to exploit academic computing resources, edge computing platforms like BOINC [2] and Seattle Open P2P Computing project [3] and applications like SETI@home [4] which seek to harness unused home computing resources. The Nano Data Centers Project [14] is an EU FP7 project to explore the notion of a distributed datacenter built out of unused edge resources like settop boxes, internet gateways and Wifi access points. The Personal Cloud concept differs from all the above. First, the Personal Cloud takes unused x86 based computers (not settop boxes) at home and converts the collection into a standard IaaS cloud hosting arbitrary VMs, rather than a restricted application level virtualization environment. Second, unlike a Grid, or a Community Cloud [5] which pools unused resources into a common Cloud, the user hosting the Personal Cloud does not do so to run VMs on his own Personal Cloud, though the user is free to do so. The goal of the Personal Cloud is to instead host VMs of others safely in the user's home network — the motivation for hosting being either access to the Personal Clouds of others, or credit in virtual or financial form. Finally, when a user runs VMs on a remote Personal Cloud, the VMs are isolated in a *Virtual Private Cloud*(VPC) [6] such that they share a common address space, which allows them to communicate directly amongst themselves, but are isolated from both the home network address space and the VPCs of other users who are running VMs on the same Personal Cloud.

We expect three distinct use cases of the Personal Cloud. The first use case, on which we focus in this paper, is for individual users to create Personal Clouds and use them to run cloud based applications such as online backup and file sharing, bypassing traditional public clouds for reasons of cost or personal preference. The second use case is to federate the individual Personal Clouds into a single distributed virtual data center cloud, at a cost that is a small fraction of traditional clouds. The final use case is in universities and enterprises, where departmental and lab resources can be packaged into indi-

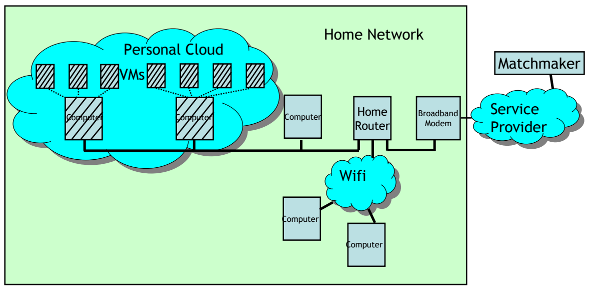

Figure 1: High Level Overview of a Personal Cloud

vidual Personal Clouds, which can then be combined to meet requirements.

The Personal Cloud lifecycle follows four steps. First, the user creates a Personal Cloud by installing the Personal Cloud software onto unused computers using standard install media like a CD or Flash Drive. Second, the user trades VMs by offering VMs from his cloud, and requesting VMs from other clouds. A key contribution of this paper is the concept of *matchmaking*, or matching the VM requests and VM offers of each user in an optimal manner such that the maximal number of requests are satisfied. Third, the VM assignments of the matchmaking algorithm are used by each user to instantiate and allocate local VMs to remote users, and to access remote VMs assigned to the user. Lastly, as participating computers, users, requests and offers change, there is an incremental change to the VM assignment. We expect all steps to be simple enough for laymen to participate in the lifecycle. The initial sections of the paper present the concept of the Personal Cloud and explain its architecture and implementation, while the later sections describe the Matchmaking algorithm and its simulation.

2 Architecture of the Personal Cloud

Assumptions A detailed view of the Personal Cloud is shown in Figure 1. VMs run on the subset of home computers that is dedicated to the Personal Cloud and are assumed to be always powered on and static. All these home computers are connected to a home network, which is in turn connected to the Internet via a Home Router and a broadband modem. The home router obtains at least one public IP address (not necessarily static), uses DHCP to distribute private IP addresses to computers in the home network, and uses NAT (Network Address Translation) to share the public IP address(es) amongst all the home computers. We assume that the vast majority of Personal Clouds will consist of a small group of computers, hence scalable intra cloud networking is not an issue. The architecture we present is hypervisor agnostic and complementary to existing cloud

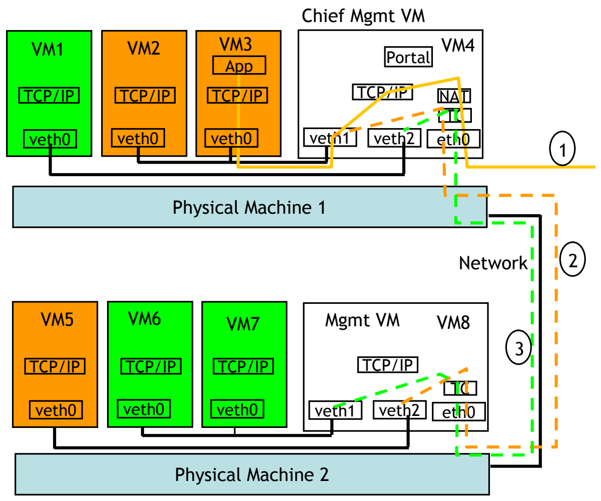

Figure 2: Virtual Networks inside the Personal Cloud

management solutions such as Amazon EC2 [7] and Eucalyptus [8]. We highlight that functionality needed for a Personal Cloud that is missing in conventional cloud management solutions.

Some of the principal technical challenges addressed in our current implementation of the Personal Cloud are: *Security and Privacy of the VMs running on the Personal Cloud*, *VM Network isolation*, *Bandwidth (BW) sharing*, *IP Address sharing and NAT Traversal*.

It is important to isolate and secure the hosting user from the VMs in the Personal Cloud and prevent malicious snooping, Denial of Service attacks or excessive bandwidth consumption from these VMs. In this paper, we focus on this issue, and show how we isolate users of the Personal Clouds into individual VPCs with capped bandwidth limits. We rely on a key VM called the *Management VM* (Mgmt VM) which is installed on each machine in the Personal Cloud. These VMs, and their internal architecture is show in Figure 2. One of the Mgmt VMs becomes the *Chief Management VM* (Chief Mgmt VM) through an automatic election process. The Mgmt VMs are lightweight enough to impose minimal overheads, and can be folded into the hypervisor for those hypervisors which provide the requisite interfaces, *e.g.*, the Dom0 domain in Xen or the Linux host in KVM.

Management The Chief Mgmt VM acts as the gateway for Internet traffic to and from the Personal Cloud. It runs a web portal which manages the Personal Cloud VMs. Amongst its tasks are Personal Cloud automation including the instantiation and deletion of VMs, assignment of VMs to users, assignment of VMs to virtual networks, isolation and security of inter VM traffic, QoS restrictions on network traffic to and from the Personal Cloud, and IP address sharing and application proxying across multiple VMs.

VM Networking The main goal of VM networking in the Personal Cloud is to provide each user of the cloud with a separate, fully isolated virtual network, while shielding the local user traffic from the Personal Cloud traffic. Unlike existing cloud networking schemes like Seattle [12] or NetLord [13] which focus on creating a scalable L2 infrastructure within a data center, our goal for VM networking is creation of a large number of virtual networks without any assistance in the form of VLANs or custom firmware or IP level forwarding from any switch or router since no such capability can be assumed for home networks. All VMs belonging to the same user are put in the same virtual network, each with an address space distinct from the home network address space. For example, Figure 2 shows VM1 to VM8 running inside a Personal Cloud. VM1, VM6 and VM7 are assigned to the same remote user, and hence are in the same virtual network. VM2, VM3 and VM5 belong to a different user, and are part of a different virtual network. Each Mgmt VM maintains a separate virtual network for each of the different users running VMs on that physical machine. Effectively, the VMs of each remote user are put in separate Virtual Private Clouds (VPCs). These Virtual networks are bridged across physical machines to the Chief Mgmt VM, so that VMs belonging to the same user, but on different physical machines can still belong to the same virtual network, and the Chief Mgmt VM is able to address all VMs in the Personal Cloud. The Mgmt VMs have a separate virtual network for management communications. Since the Chief Mgmt VM is connected to all virtual networks, it assigns IP addresses out of a private address space to each VM via DHCP. Each virtual network is disjoint, so even if the same address range is used for different virtual networks, there is no network address collision.

Figure 2 also shows several different traffic flows. Traffic flow 1 is traffic from VM3 to the Internet. Since the Chief Mgmt VM acts as the default router for the Personal Cloud, all exiting traffic is routed via it. Here the traffic is NATed to the address assigned by the home router and is forwarded to the home router (not shown) for transmission to the Internet. Traffic flows 2 and 3 represent L2 tunnels maintained between the two Mgmt VMs to forward traffic belonging to the two virtual networks in the Personal Cloud.

The TC box in each Mgmt VM represents the *Traffic Conditioner*. This is used to rate limit the traffic from each VM, in order to regulate the BW usage both inside the home network, and into the Internet. Note that while the NAT module is only present in the Chief Mgmt VM, the TC module is present in all Mgmt VMs, since it is necessary to regulate the Personal Cloud traffic not only to and from the Internet, but also within the home network.

Address Sharing and NAT Traversal One major networking issue with Personal Clouds is the sharing of the limited public address space (usually just one) with the multiplicity of VMs such that each remote user can access his or her VM independent of other users. Note that the problem lies with traffic coming from the Internet to the Personal Cloud. Outgoing traffic from the Personal Cloud is NATed as normal by the home router, and does not have any reachability problems. It is possible to create a reverse tunnel from the VMs to a public IP address to allow external access, and this is the usual solution for external access to NATed systems. However, this can lead to inefficient triangular routing, and therefore, we incorporate an optional, innovative, two stage approach to providing authenticated access to individual VMs inside the Personal Cloud. Our solution to managing incoming traffic is to forward incoming Internet traffic to the Chief Mgmt VM, which then uses its web portal to map incoming traffic to the appropriate VM. In the first stage, the remote user accesses the web portal in the chief Mgmt VM over the Internet. After suitable authentication, the user is then directed to a set of links representing the VMs that the remote user is running in the Personal Cloud. By clicking on a particular link and selecting the appropriate protocol type, subsequent user traffic of that particular protocol type is redirected to the selected VM. This technique provides an authenticated, safe mechanism for direct inbound access to VMs from arbitrary IP addresses without any triangular routing.

Instead of this two stage approach, if each VM were to be accessed only via the web, then the web portal in the Chief Mgmt VM can run as a web router. This is because each HTTP request uniquely identifies the DNS hostname it is addressed to. Since each VM can have a different DNS name while sharing the same IP address, this allows for a one stage demultiplexing at the portal.

3 Matchmaking Problem and Algorithm

A key aspect of the Personal Cloud is the matchmaking or resource sharing aspect, in which individual users express the number of VMs they are willing to host in their Personal Cloud, and the number of VMs they wish to use in other Personal Clouds. We assume that the matchmaking algorithm runs in a central entity called the Matchmaker to which individual Personal Clouds submit their individual requests and offers. Note that the matchmaking algorithm presented can be used for *any* edge computing scheme which relies on P2P resource sharing, such as BOINC. Also note that the Matchmaker functionality *only* does VM assignments and can be run on one of the Personal Clouds. The P2P nature of the Personal Cloud, the requirement that a given user's VM request not be satisfied by its own VM offer and the long term nature of the assignment precludes the use of existing VM assignment solutions. Conventional VM assignment and provisioning solutions either rely on bin packing algorithms, or focus on the challenges in assigning short term jobs to VMs in a conventional data center [9] or focus on policy based scheduling, *e.g.*, energy efficient VM assignment [10]. In this section, we present a mathematical formulation of the VM resource sharing problem, and an algorithm for finding optimal resource assignments. Due to space considerations, we omit most formal details and proofs.

Specification of Offers and Requests While we apply our algorithm for the purpose of sharing VMs, our formulation of the resource sharing problem is agnostic to the kind of resource that is shared (*e.g.*, CPU, bandwidth, storage or VM) — we simply assume that there is a "resource unit" standing for some granularity at which any size/amount of interest of that resource can be represented as a discrete multiple of such resource units. We then assume that each node in the personal cloud is capable of "hosting" some number of such resource units and requests a certain number of such resource units to be hosted at other nodes. The resource units requested by each node may be distributed across the other nodes in the cloud. We allow the requesting node to constrain the distribution by specifying, for each remote node, an upper bound on the number of resource units placed at that remote node in any acceptable distribution.

Following the above intuitions, an instance of the *matchmaking problem* $\mathscr{M}$ is given as $(n, \mathbf{O}, \mathbf{R}, \mathbf{U})$, with n specifying the number of peer nodes in the Personal Cloud, $\mathbf{O}$ specifying an offer vector with $\mathbf{O}(i)$ giving the total number of resource units that node i is willing to host (on behalf of all other nodes), and $\mathbf{R}$ specifying the request vector with $\mathbf{R}(i)$ giving the total number of resource units that node i desires to be hosted (across all remote nodes). Finally, $\mathbf{U}$ is the upper-bound matrix with $\mathbf{U}(i,j)$ giving the maximum number of resource units that node i is willing to have hosted (on its behalf) at node j.

Upper bound constraints are general enough to specify many resource sharing scenarios of interest. For example, allowing an arbitrary distribution of a request across remote nodes can be specified by using upper bounds of infinity. Specific peer nodes can be disallowed from hosting a request by specifying an upper bound of 0 for them. Finally, fault-tolerant replication so that replicas are hosted at distinct remote nodes can be achieved by specifying an upper bound that is the size of the replica for acceptable remote nodes (*e.g.*, those that are at a sufficiently large geographic distance from the requesting node) and an upper bound of 0 for all other nodes.

Matchmaking Problem We are interested in finding

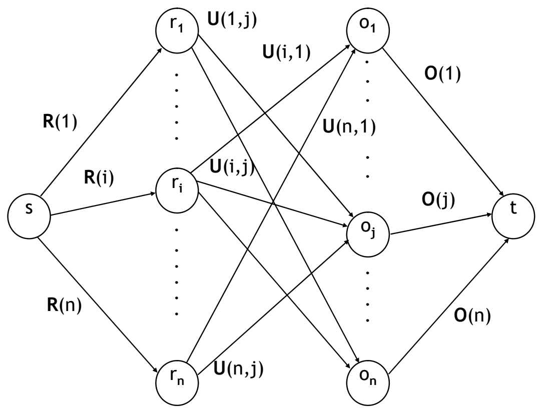

Figure 3: Reduction of the Matchmaking problem to the Maximum flow in a graph

an allocation of resource requests across the nodes in the Personal Cloud. We call such an allocation a *matchmaking assignment* and represent it as a matrix $\mathbf{A}$ with the value $\mathbf{A}(i,j)$ giving the number of resource units of node i that are hosted on node j. For an instance of the matchmaking problem $\mathscr{M} = (n, \mathbf{O}, \mathbf{R}, \mathbf{U})$, a matchmaking assignment $\mathbf{A}$ is said to be *feasible* if it respects the constraints embodied in the instance. Specifically: (a) the total number of resource units assigned by $\mathbf{A}$ to any node i must not exceed the corresponding offer $\mathbf{O}(i)$, (b) the total number of units hosted on behalf of any node i must not exceed its corresponding request $\mathbf{R}(i)$, and (c) the upper-bound constraints must be respected, *i.e.*, $\mathbf{A}(i,j) \leq \mathbf{U}(i,j)$ for all i, j.

Define the *cumulative request-assignment* of a matchmaking assignment $\mathbf{A}$ to be the total number of units hosted on behalf of all nodes. A *request-optimal* assignment $\mathbf{A}$ for an instance $\mathscr{M}$ of the matchmaking problem is a feasible assignment for $\mathscr{M}$ whose cumulative request-assignment is maximum among all feasible assignments. A matchmaking assignment is *request-satisfying* if each node's request is fully met by the assignment. It can be shown that if an instance of the matchmaking problem admits a feasible request-satsifying assignment then a request-optimal assignment for it has to be request-satisfying. Furthermore, while some instances of the matchmaking problem may have no request-satisfying assignments, all instances have a request-optimal assignment. We therefore consider the matchmaking problem to be the more general optimization problem of computing request-optimal assignments.

Definition 1 (Matchmaking Problem)**.** *Given an instance of the matchmaking problem $\mathscr{M}$, compute a request-optimal assignment $\mathbf{A}$ for $\mathscr{M}$.*

Optimal Matchmaking Algorithm Our algorithm for the Matchmaking Problem (Definition 1) is based on a reduction to the maximum flow problem. Specifically, for any instance $\mathscr{M} = (n, \mathbf{O}, \mathbf{R}, \mathbf{U})$ of the matchmaking problem, we construct the flow-graph $G(\mathscr{M})$ illustrated in Figure 3. In addition to a source node s and a sink node t, it has "request" nodes $r_1, \ldots, r_n$ and "offer" nodes $o_1, \ldots, o_n$. There are edges from s to each of the request nodes with capacities taken to be their corresponding requests. There are cross-edges from request nodes to offer nodes except when they correspond to the same node (reflecting no self-hosting) and have capacities given by the upper-bound constraints. Finally, there are edges from the offer nodes to the sink node with capacities taken to be their corresponding offers.

We can establish a correspondence between feasible assignments for the matchmaking instance $\mathscr{M}$ and integral flows in the constructed flow-graph $G(\mathscr{M})$. Specifically, for any integral flow f in the flow-graph, we can construct a corresponding matchmaking assignment $\mathbf{A}(f)$ by using the values of the flow f on the cross-edges. We can show that the assignment $\mathbf{A}(f)$ is feasible if and only if f is a valid flow, and that the cumulative request-assignment of $\mathbf{A}(f)$ is exactly equal to the value of the flow f. Finally, the flow-graph $G(\mathscr{M})$ has the property that all capacities are integral. A well-known result on maximum flow is that for any graph with integral capacities, there is a maximum flow that is integral. Consequently, we can establish the following reduction of the Matchmaking Problem to the maximum flow problem.

Theorem 1. *Let $\mathscr{M} = (n, \mathbf{O}, \mathbf{R}, \mathbf{U})$ be an instance of the matchmaking problem and f be an integral maximum flow for $G(\mathscr{M})$. Then $\mathbf{A}(f)$ is a request-optimal assignment for $\mathscr{M}$.*

Our algorithm for computing optimal matchmaking assignments, based on Theorem 1, is then as follows. For any input instance $\mathscr{M}$ of the matchmaking problem, we construct the flow graph $G(\mathscr{M})$, compute its maximum flow f, and return the matchmaking assignment $\mathbf{A}(f)$. The construction of the graph $G(\mathscr{M})$ takes time linear in the size of the input $\mathscr{M}$ with number of vertices being $O(|\mathscr{M}|^{1/2})$ and number of edges being $O(|\mathscr{M}|)$. The maximum flow can be computed using any standard maximum flow algorithm such as Edmunds-Karp in polynomial time. It is also possible to extend the resource sharing algorithm to allow hosts to charge per resource costs, and to incrementally adjust optimal assignments.

4 Simulation Results

We have extensively simulated the Max Flow (MF) reduction of the Matchmaking algorithm described in the previous section using an off the shelf MF implementation, while baselining it with a Greedy (GR) version of the Matchmaking algorithm. The Greedy version simply runs linearly through the set of requests, allocating the current request from the first available offer. The simulations vary both the offers and requests of each user, as well as the number of users. The key result is that the MF algorithm consistently outperforms the GR algorithm, especially when the size of individual requests is large compared to the total number of users. Our definition of the Matchmaking algorithm is general enough to allow many different semantics to be associated with the sharing of resources. For example, a request for 3 VMs can either be a request for 3 unique replicas, one per Personal Cloud, or allow for arbitrary distribution, in which multiple VMs can be located in a single Personal Cloud. We refer to the first case as the Replicated Request (R) case, and the second as the Unconstrained Request (U) case. We plot the performance of both the U and the R versions for both the MF and GR algorithms.

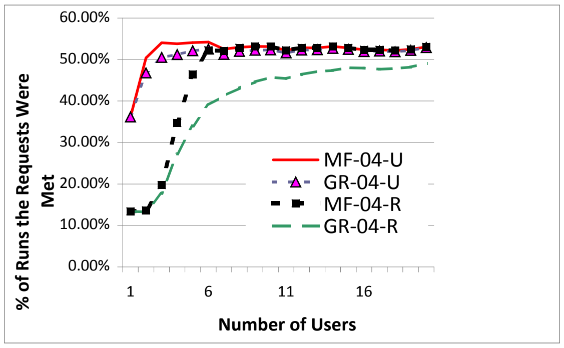

Figure 4: Percentage of total request met by the Max Flow and the Greedy Algorithms as the number of users are varied— Requests between 0 and 4

Figure 4 plots the percentage of the total request met as the number of users are varied. Due to the convergence of results as the number of users increases beyond 20, we only show the results for up to this many users. The cases considered are the Max Flow Unconstrained case (MF-04-U), the Max Flow Replicated case (MF-04-R), the Greedy Unconstrained case (GR-04-U) and the Greedy Replicated case (GR-04-R). Given that the requests and the offers are randomly generated, we expect that in 50% of the cases, the requests cannot be satisfied.

As the number of users increases, the percentage of total request that is met approaches the limiting value of 50% for the MF versions. We see that in this scenario, where the requests and offers are comparable to the number of users, once the number of users increases past 4, the MF versions quickly converge to the 50% limit, while the GR versions take considerably more users to

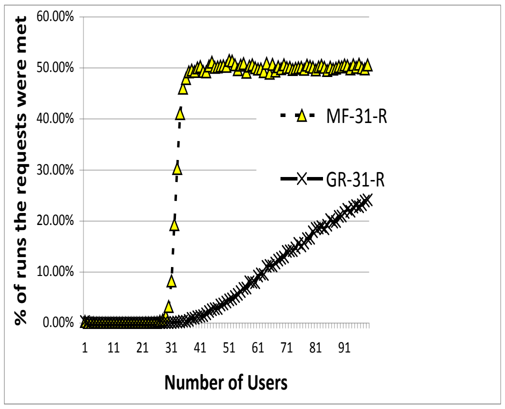

Figure 5: Percentage of total request met by the Max Flow and the Greedy Replicated Algorithms as the number of users are varied— Requests between 0 and 31

approach the limit. Also seen is that the *U* case always achieves a higher success rate than the *R* case for both the MF and GR version. This is as expected, since the *R* case requires each request to be replicated uniquely, but that is not possible if the number of users is small compared to the individual requests.

When individual requests are large compared to the number of users, it is expected that a naive approach like GR would underperform the MF algorithm. We examine cases where the individual requests are for for upto 31 VMs, rather than for upto 4 VMs. This is plotted in Figure 5, which shows the percentage of runs requests were met for the GR and MF replicated versions. As expected, until the number of users exceeds 31, both the GR and the MF versions are unable to satify the requests using unique replicas, but once the number of users crosses 31, the MF version is quickly able to reach the expected 50% match of requests, while the GR version is unable to converge to this value even with a hundred users.

5 Conclusion

In contrast to existing application sharing and virtualization approaches, the Personal Cloud model builds an IaaS cloud that provides the necessary NAT , Virtual Networking and Traffic Conditioning support needed to run VMs securely inside the home network. By building the Personal Cloud and by simulating the Matchmaking algorithms, we have verified that both combined provide a scalable and viable technique for sharing edge compute resources on the Internet.

There still exist many challenges involved in moving from the prototype to the operational stage. One open research issue (applicable for any cloud deployment not just the Personal Cloud) is securing the cloud user from the cloud provider. It is important to create a secure, tamperproof environment for the VMs in the Personal Cloud which prevents the hosting provider from snooping on the contents of the VMs' memory, storage or network traffic. We are exploring a tamperproof VM hosting framework based on secure bootloading of hypervisors and VMs for memory protection, encrypted filesystems and tunnels for disk and traffic protection, and distributed task and storage scheduling to limit exposure from security breaches. Given the consumer grade equipment involved, providing reliability is another open issue. Existing commercial cloud deployments based on off the shelf equipment, such as the Google File system [11], still rely on carrier grade data centers and networking, both of which are absent in the Personal Cloud. Finally, it would be necessary to adapt existing cloud middleware, like MapReduce/Hadoop to the heterogeneous distributed environment of the Personal Cloud, so that the Personal Cloud can run not just raw VMs, but also standard cloud middleware based services.

References

[1] L. Peterson, A. Bavier, M. Fiuczynski, and S. Muir, "Experiences implemeting planetlab," in *OSDI*, 2006.

[2] D. Anderson, "Boinc: A system for public-resource computing and storage," in *5th IEEE/ACM International Workshop on Grid Computing*, 2004.

[3] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: a platform for educational cloud computing," in *Proceedings of the 40th ACM technical symposium on Computer science education*, 2009.

[4] D. Anderson, "Seti@home: An experiment in public-resource computing," in *communications of the ACM, Vol. 45*, 2002.

[5] G. Briscoe and A. Marinos, "Digital ecosystems in the clouds: towards community cloud computing," in *IEEE DEST*, 2009.

[6] T. Wood, P. Shenoy, A. Gerber, K. Ramakrishnan, and J. V. der Merwe, "The case for enterprise-ready virtual private clouds," in *Usenix HotCloud*, 2009.

[7] [Online]. Available: http://aws.amazon.com/ec2/

[8] [Online]. Available: http://open.eucalyptus.com/

[9] A. Gulati, G. Shanmuganathan, A. Holler, and I. Ahmad, "Cloud scale resource management: Challenges and techniques," in *HotCloud*, 2011.

[10] C. Lin, P. Liu, and J. Wu, "Energy-efficient virtual machine provision algorithms for cloud systems," in *Fourth IEEE International Conference on Utility and Cloud Computing*, 2011.

[11] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SOSP*, 2003.

[12] C. Kim, M. Caesar, and J. Rexford, "Floodless in seattle: A scalable ethernet architecture for large enterprises," in *Sigcomm*, 2008.

[13] J. Mudigonda, P. Yalagandula, J. C. Mogul, B. Stiekes, and Y. Pouffary, "Netlord: A scalable multi-tenant network architecture for virtualized datacenters," in *Sigcomm*, 2011.

[14] [Online]. Available: http://www.nanodatacenters.eu/