USENIX Association

Proceedings of HotOS IX: The 9th Workshop on Hot Topics in Operating Systems

Lihue, Hawaii, USA
May 18–21, 2003

Virtual Appliances in the Collective: A Road to Hassle-Free Computing

Constantine Sapuntzakis and Monica S. Lam
Computer Systems Laboratory
Stanford University

Abstract

This paper describes the vision of the Collective, a compute utility which runs internet services as well as the highly interactive applications we run on desktop computers today. As part of this vision, we wish to shift the burden of administering the desktops from users to professionals. To decrease the cost of administering systems, we find inspiration in the reliability and maintainability of network-connected computer appliances. We argue for structuring our software as a group of networked appliances, each appliance virtualized on a virtual machine monitor. We show how to run virtual appliances in the Collective system and examine some ways in which individuals and groups may adopt virtual appliances.

1 Introduction

The progress of computer hardware has given us abundant computation, storage, and communication capacity. Our only limitation seems to be our ability to use and manage this wealth. The challenge is to develop software that makes compute resources into a utility as easy to use as water, power, and the telephone.

Users expect the following from a compute utility:

1. *Global, uniform access.* Users should have access to their computing environment anywhere in the world, as if they were in their own offices. The computing environment should include not just web services but all the applications people run on their computers today.

2. *Hassle-free computing.* Users today spend too much time administering their machines, that is, if they know how. After all, many home computer users do not backup their systems or apply the latest security patches. In a utility, computer administration must disappear into the infrastructure, becoming invisible to users.

3. *An open system.* To encourage competition in delivering robust, easy-to-use software systems, a utility must not require that applications be run on a particular operating system or be written in a specific programming language.

4. *Security in a public infrastructure.* A user would like the computers in the utility to perform computations correctly on their behalf, to respect their privacy, and to not corrupt their data. We foresee that a combination of laws, such as those against interfering with the mail, and technologies, like frequent backups, trusted computing, intrusion detection, and auditing, will give users enough confidence to move their computation to a utility. Trust is a matter of extent. Just like with the credit card system, users will not expect the utility to be completely trustworthy. A company may choose to operate on its sensitive data only on the company's internal compute utility.

Today's computing environments are a far cry from the compute utility described above. This paper presents the high-level design of our Collective system architecture, an attempt to create this utility. As a first step, we look at ways to make computing environments more reliable and easier to manage. Inspired by computer appliances, which trade off generality for ease of use and reliability, we propose to structure our computing environments as collections of appliances. By virtualizing the appliances, multiple appliances can run on one piece of hardware, making the model affordable for a far wider range of software and users.

This paper describes how the ideas in the Collective can change our computing landscape. It makes the following main points. First, we describe the vision of a future computing environment where compute services are regarded as a utility. Second, we argue for splitting a computer's software into multiple virtual appliances. Third, we briefly describe the functions of the Collective system software. Fourth,

we discuss the socio-economic ramifications of this proposed architecture. Although the Collective is designed with the goal of creating a global utility, certain ideas, like virtual appliances, can be readily adopted to solve today's software management problems.

2 Design Concept

Because PCs are hard to use and manage, some have predicted that appliances will become popular [13]. This section describes how we can borrow ideas from the design of appliances to improve the manageability and usability of computers.

2.1 Appliances

With the falling cost of computer hardware, special-purpose appliances abound, e.g. firewalls, VPN gateways, game consoles, TiVos, and NetApp filers [7]. While some of these appliances are built out of PC hardware and run PC operating systems, appliances differ from PCs in several ways. For one, an appliance does not try to do everything and, as a result, is easier to use and maintain. The appliance comes with the software needed to serve its purpose. The appliance maker tests all the software to ensure it works together; on the PC, the user is often mating an application with a version of the operating system or libraries that it was never tested with. Finally, two appliances are better isolated than two application installed on the same operating system, reducing the chance that the bad behavior of one will harm the other.

There is one more, perhaps most significant, difference: appliances, especially networked ones, are maintained by the makers, not the users. On an appliance, the maker controls all the software and can create correct updates with higher confidence. Network-connected appliances such as the TiVo download updates periodically to fix bugs, add new features, and plug security holes. In contrast, on Windows and Linux, users must initiate software updates; makers are nervous about breaking a user's configuration.

2.2 Using Appliances to Structure the Computing Environment

To gain ease of use and manageability, we can structure our computing environments as groups of appliances. A user might have an appliance for each application he uses today, for example, an AOL appliance, an office suite appliance, and a video editing appliance. A user may even have multiple appliances with similar software: an office suite appliance for work, and an office suite appliance for personal correspondence. A user may wish to bundle multiple appliances into a single unit. For example, a company might want to make sure that a telecommuter's office suite appliance is protected by the company's firewall/VPN appliance and audited for break-ins by an intrusion detection appliance.

By placing proxy appliances at the network ports of a current appliance, we can roll out new network protocols without modifying currently running appliances. IPsec can be deployed in an encryption/authentication appliance. A new network file system can be deployed with a translator to and from NFS. The proxies can be implemented at user level as packet filters without worrying about deadlocks.

Each appliance is connected to the network and maintained by the maker. The user extends their environment by getting more appliances. Network protocols can be used to promote sharing between appliances, like network cut-and-paste[11] and shared network file systems for user files.

Still, hardware appliances have their limitations. Hardware is expensive realtive to software, takes space, power, generates noise and heat, and must be physically delivered. An appliance's hardware can fail, potentially trapping configuration and user state, making it hard to recover.

2.3 Virtual Appliances

Many of the limits described in the previous section can be overcome by making appliances virtual. A virtual appliance is the state of a real appliance (the contents of the appliance's disks) as well as a description of the hardware (e.g. two Ethernet adapters, 256mb RAM, two hard disks, etc.). Since a virtual appliance is just data, it can be shipped electronically. A virtual appliance runs in a virtual machine monitor (VMM), sits on a virtual network, and stores data on virtual disks or network storage. The virtual appliance talks over the network to real I/O devices, like displays, printers, game pads, and keyboards.

Using a virtual machine monitor (VMM), like VMware GSX server[16], we can run many virtual appliances on a single computer, spreading the cost, power, space, and heat over multiple appliances. This will make the appliance model affordable for a wider range of applications.

We can run the same software that was on the hardware appliance. Since the VMM hides differences in the physical hardware, the appliance maker can maintain a small set of device drivers and still have its appliances run on wide range of hardware. To ease the

transition to a networked world, the VMM can map the virtual appliance's hardware devices and protocols to network devices and protocols. While a virtual appliance may think that its computer has a hardware display adapter, the virtual display will actually talk over the network to a remote display on a thin client. A virtual appliance may think that it is talking to a local hard disk, but instead the hard disk is hosted on a reliable network storage service.

3 Collective: A Network of Hosts of Virtual Appliances

In this section, we describe the Collective architecture. The Collective is a compute utility based on the concept of appliances. In the Collective system, machines serve as caches of virtual appliances, and the states of appliances are saved in some persistent data store.

The Collective software uses the VMware x86 virtual machine to execute, resume, and suspend virtual appliances. There are many advantages to virtualizing the x86 architecture. The machine can run any virtual appliance that runs on x86 hardware, a de facto standard. It does not require the software to run on any particular operating system. Because the operating system is included in the appliance, system administration is performed by the appliance makers, not the users. Finally, because the VMware virtual machine monitor is a commercial product, we can run experiments on a usable prototype system.

The computers that run virtual appliances are called hosts. To create a utility out of these hosts, the Collective software provides the following additional functions:

1. *Virtual networks of virtual appliances*. To implement a network of virtual appliances, not only does the Collective provide a virtual machine interface, it also virtualizes the network. It uses (1) Ethernet virtual LANs (or VLANs) to connect virtual appliances on separate physical hosts, and (2) virtual Ethernet switches on the same machine to create multiple isolated networks within a single host.

2. *A networked service plane*. The Collective provides a "service plane" that automates the management of virtual appliances and hardware resources. The Collective keeps the virtual appliances up to date, replicates them, migrates them as needed to present the user with the illusion that he has instantaneous and fast access to all the latest appliances wherever he goes. This is a challenge since the state of an x86 machine can be large. Our previous work proposed various optimizations to enable new appliances to start up quickly, to reduce the amount of traffic needed to update an appliance, and to speed up the migration of appliance states between machines[14]. The service plane will also perform optimizations such as load balancing to increase the utilization of hosts in the system.

3. *Introspective facilities.* Having access to the state in appliances and running as a separate entity, the Collective can provide introspective services to add features to appliances. For example, the Collective can examine the state of an executing appliance to detect signs of intrusion[4]. This is superior to implementing intrusion detection in the appliance because the detector itself would have to guard against being compromised. Another example is a general checkpointing facility for error recovery. The Collective can checkpoint the state of an appliance as it executes so that users can access a prior appliance state should an error occur.

4. *A trusted computing platform.* Before we run a job on a machine in the utility, how can we tell that the software on the host is not malicious and will respect our security and privacy concerns? One option is to ensure the host is running a trusted virtual machine monitor[5], attested to by tamper-proof hardware.

The above sketches our high-level approaches towards providing the properties desired of a utility, as described in Section 1. The service plane of the system migrates appliances efficiently to give users global access to their computing environment. The concept of actively managed virtual appliances reduces the hassles in computing. Openness is achieved by adopting the x86 architecture interface. Finally, ideas like TCPA are used to provide some degree of trust in the infrastructure.

4 The Socio-Economic Landscape

Creating a global utility is an ambitious goal. Not only are there many technical details to work out, it is important that economic incentives be in place to make such a system happen. It is important that we can stage the development by creating subsystems that address some of the real problems we face today. In this way, we can gain valuable experience needed

to build the ultimate system. In the following, we describe how a subset of the ideas described above can be used to solve real problems encountered by today's computer users both at homes and in the workplace.

4.1 Information Technology for Organizations

Manpower, not hardware cost, dominates the information technology (IT) spending in many large organizations. IT departments provision one system administrator for every 20 to 50 computers.

The Collective will make it easier to deploy and maintain turnkey solutions for many markets, decreasing the IT staff, and perhaps eliminating it in many small and medium-sized organizations. Organizations which benefit from unique IT processes or require special applications will continue to retain an IT staff, much like they retain one today to deploy applications and services. For example, an IT department may make a special bundle of appliances for finance which is different from the bundle for human resources. Professors may have a set of appliances for their research and another set for sharing with students.

The Collective can also ease these IT tasks:

1. *Unauthorized applications*. The IT department can provide a set of working, core appliances firewalled from the rest of a user's setup. Even if a user adds other, unauthorized or untested appliances, the isolation and firewalling of the appliance model should keep the core working, even though the new appliances may break or misbehave. If, after experimenting with the appliance in isolation, the appliance turns out to be useful, the IT department can give it more access to other core appliances and data or even make it part of the core.

2. *Setting up new offices*. We can set up a new office quickly by simply buying the additional hardware and connecting it to the network. Complete appliances from the organization's headquarters or purchased from third-party companies will automatically populate the machines as employees use their appliances.

3. *Administration of branch offices*. Some organizations have many branch and sales offices dispersed geographically. With our architecture, there is no difference between the employee's experience in the headquarters or the branch offices, because the relevant appliances in all the offices are updated automatically.

4. *Relocation and telecommuting*. With our architecture, a user can access their running appliances from any machine. The system will automatically migrate and cache the appliances to give users fast response. With this feature, employees can work at home or move between offices without worrying about moving files or restarting applications.

5. *Error and disaster recovery*. All of the appliances can be backed up at a remote location and retrieved if errors are discovered or disaster strikes. In addition, our system's introspective ability can be used to save the active state of an appliance as it executes. This is useful for recovering from errors not just in the software and hardware, but also from operator errors.

4.2 Home Users

It is ironic that professionals in enterprises and universities are supported by system administrators, whereas novice home users are not. How do we expect novice users to know about backups, apply security patches, and run virus detectors and disk defragmenters? We believe that plenty of frustrated home users will gladly move to an easy-to-use and maintenance-free collection of virtual appliances.

Instead of buying a PC, consumers would buy a "universal appliance host" which bundles the x86 processor with a thin layer of software, including a VMM. User files will be managed by a storage service, which keeps the user's data locally and as well as encrypted backups at a remote site.

We expect that there will be many companies who specialize in developing attractive easy-to-use virtual appliances for each market segment. (This model does not preclude the development of free and personal appliances.) There will be appliances for senior citizens, novice users, hobbyists of different kinds, teenage girls, teenage boys, and children of different ages. Each of these appliances will combine a large number of software titles, some of which may have been developed by a third party. There will be more titles than any individual would typically install on his machine. Or, if the features in a single appliance do not suffice, people may run multiple appliances.

Users will rent or subscribe to the appliances; using individual software titles in the appliance may have additional fees. In return for paying a fee, appliances will be actively maintained and updated by these companies. The predictable update model provided by the Collective will keep the costs of the service low.

4.3 Large-Scale Services

It is interesting to compare the proposed software model with services provided by portals such as AOL and Yahoo. These portals give their customers services such as email, browsing and instant messaging. These services are kept up-to-date; users get the benefit of new features such as spam filters, virus scanners, and parental control, without having to modify their own machines. And, users can get access to these services anywhere they go.

Portal computing has its disadvantages. To serve a large number of clients, portals offer services that are not too computationally demanding. With a global utility like the Collective, there can be a continuum between central portal services and distributed user appliances. Using the Collective, the central service can replicate itself to handle load or improve interactivity. Depending on how state is shared, the service can choose to partition state across the replicas. Desktop appliances are the extreme. Since there is little sharing, the service is partitioned to handle only one user's session; the service is placed at the user's computer for good interactivity.

Today, data is trapped at the portal, making it harder to use across portals. For the convenience of not having to manage software, many users have chosen to entrust their private data to unknown companies that may go under, selling or destroying their data. In the Collective, users will bring their own storage to the service.

5 Related Work

Sun's N1[15], IBM's "autonomic computing"[1], Duke's "Cluster-on-Demand"[12], and HP "utility data center"[8] all aim to simplify the mapping of services onto pools of computers, networks, and storage. Grid computing aims to provide a large pool of computing for scientific applications[3]. In the Collective, we aim to manage not just web services and scientific applications but highly interactive applications that reside today on the desktop.

Goldberg surveyed the field of virtual machines[6]. More recently, Disco revitalized interests in virtual machines; a major change from the previous work had been the development of computer networks and network protocols to share data easily and quickly between multiple computers[2]. Internet suspend/resume describes how to use the VMware VMM to provide user mobility in the wide area[10]. The Collective uses the VMware VMM in the same fashion to provide user mobility[14]. The Denali isolation kernel shows that with a couple of minor architectural modifications, it is possible to scale up to hundreds of virtual machines on a single computer[17].

Java provides mobile code in a portable virtual machine but requires users to rewrite their code in Java and to new interfaces[9]. In contrast, virtualizing using a VMM allows us to use the large amounts of code already written to run on today's hardware platforms, including Java.

6 Summary

By structuring software as a group of network-connected appliances, users will manage software less. With appliances, the maker controls the software installed on the appliance, allowing the maker to update the software without user intervention and with predictable results. Users can still extend their environments by adding appliances. Virtualizing appliances makes them cheaper and more manageable. This allows us to apply the appliance concept to more users and more applications.

While an end user or small business will likely subscribe to appliances that suit their needs and tastes, large companies will continue to have an in-house IT department, which will use appliances to manage the company's computing assets.

A Collective-like global utility enables software to be replicated, partitioned, and pushed into the network by encapsulating it in appliances, forming a continuum between central portal services and distributed desktop applications.

7 Acknowledgements

This work was funded in part by the National Science Foundation under Grant No. 0121481 and Stanford Graduate Fellowships. We thank Ramesh Chandra and Mendel Rosenblum for discussions on the ideas of this paper.

References

[1] Autonomic computing. `http://www.research.ibm.com/autonomic/`.

[2] E. Bugnion, S. Devine, and M. Rosenblum. Disco: Running commodity operating systems on scalable multiprocessors. *ACM Transactions on Computer Systems*, 15(4):412–447, November 1997.

[3] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid services for distributed system integration. *IEEE Computer*, 35(6):37–46, 2002.

[4] T. Garfinkel and M. Rosenblum. A virtual machine introspection based architecture for intrusion detection. In *Proceedings of the Internet Society's 2003 Symposium on Network and Distributed Systems Security*, February 2003.

[5] T. Garfinkel, M. Rosenblum, and D. Boneh. A broader vision of trusted computing. In *Proceedings of the Ninth Workshop on Hot Topics in Operating System*, May 2003.

[6] R. P. Goldberg. Survey of virtual machine research. *Computer*, 7(6):34–45, June 1974.

[7] D. Hitz. A storage networking appliance. Technical Report TR3001, Network Appliance, Inc., October 2000.

[8] HP utility data center. `http://www.hp.com/solutions1/infrastructure/solutions/utilitydata/`.

[9] B. Joy, G. Steele, J. Gosling, and G. Bracha. *The Java Language Specification*. Addison-Wesley, 2000.

[10] M. Kozuch and M. Satyanarayanan. Internet suspend/resume. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 40–46, June 2002.

[11] R. Miller and B. Myers. Synchronizing clipboards of multiple computers. In *Proceedings of the Twelfth Symposium on User Interface Software and Technology*, pages 65–66, November 1999.

[12] J. Moore and J. Chase. Cluster on demand. Technical report, Duke University, May 2002.

[13] D. Norman. *The Invisible Computer*. MIT Press, 1998.

[14] C. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. Lam, and M. Rosenblum. Optimizing the migration of virtual computers. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, pages 377–390, December 2002.

[15] Sun N1. `http://wwws.sun.com/software/solutions/n1/index.html`.

[16] "GSX server", white paper. `http://www.vmware.com/pdf/gsx_whitepaper.pdf`.

[17] A. Whitaker, M. Shaw, and S. Gribble. Scale and performance in the denali isolation kernel. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation*, pages 195–210, December 2002.