Archer: A Community Distributed Computing Infrastructure for Computer Architecture Research and Education

Renato J. Figueiredo[1], P. Oscar Boykin[1], José A.B. Fortes[1], Tao Li[1], Jie-Kwon Peir[1], David Wolinsky[1], Lizy K. John[2], David R. Kaeli[3], David J. Lilja[4], Sally A. McKee[5], Gokhan Memik[6], Alain Roy[7], and Gary S. Tyson[8]

[1] University of Florida, Gainesville, FL, USA
[2] University of Texas, Austin, TX, USA
[3] Northeastern University, Boston, MA, USA
[4] University of Minnesota, Twin Cities, MN, USA
[5] Cornell University, Ithaca, NY, USA
[6] Northwestern University, Evanston, IL, USA
[7] University of Wisconsin, Madison, WI, USA
[8] Florida State University, Tallahassee, FL, USA
http://archer-project.org

Abstract. This paper introduces Archer, a community-based computing infrastructure supporting computer architecture research and education. The Archer system builds on virtualization techniques to provide a collaborative environment that facilitates sharing of computational resources and data among users. It integrates batch scheduling middleware to deliver high-throughput computing services aggregated from resources distributed across wide-area networks and owned by different participating entities in a seamless manner. The paper discusses the motivations that have led to the design of Archer, describes its core middleware components, and presents an analysis of the functionality and performance of the first wide-area deployment of Archer running a representative computer architecture simulation workload.

Keywords: virtualization, computer architecture, simulation, collaborative environments, Grid computing.

1 Introduction

Modern computer architecture research is driven by quantitative analysis. Leading-edge research requires detailed, cycle-accurate evaluation of many benchmark applications with several simulated configurations and is thus tightly dependent on the availability of high-throughput computing (HTC) systems. Many research groups are hindered in their ability to perform research because of lack of access to such resources. This is because, in addition to hardware costs, the investment of time and funds to train and educate students and staff to deploy,

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 70–84, 2009.

maintain and effectively use such systems presents a significant barrier of entry, especially for small- to medium-sized research groups. This paper describes Archer, a community-based computing resource for computer architecture research and education. Archer integrates technologies for resource virtualization, batch job schedulers, and multi-institution collaboration, in order to create:

- *A computing infrastructure which scales in capacity with community buy-in*: Archer starts from a seed set of cluster resources deployed at the Florida State University, Northeastern University, University of Texas at Austin, Northwestern University, University of Minnesota, Cornell University, and University of Florida. In addition to this seed set, each new user joining Archer with one or more desktops or servers seamlessly contribute to its aggregate capacity.
- *A system that is easy for non-experts to join and use*: Archer relies on packaging and distribution of high-throughput computing software environments as self-configuring virtual networks of virtual machine (VM) "appliances". System virtual machines [19] (such as VMware, VirtualBox) are easily installed by individual users in their own resources and can coexist with existing software in a non-intrusive manner. VM appliances allow packaging of complex system software in images that are easy to deploy [18,21]. Surveys shows that users with no prior experience with VMs can typically install and use the appliances upon which Archer is built within 30 minutes.
- *A community-based repository of simulation environments*: Archer allows sharing not only of hardware resources, but also of full-fledged software simulation modules consisting of application executables, support scripts, input and output data sets, and usage documents. In doing so, Archer facilitates the dissemination of useful tools and data sets, and foster creation of reproducible simulation experiments.

The system architecture and design choices made in Archer significantly differentiate it from related infrastructures such as the Open Science Grid (OSG) and TeraGrid, in three important ways. First, Archer enables seamless addition of resources by the community, at a fine grain (at a minimum a single desktop computer by an individual user), within minutes. This is in contrast to OSG and TeraGrid, where individual resources cannot be easily incorporated. Second, Archer deployments are virtualized and can be easily replicated, both at a smaller scale within an institution, and at a multi-institution scale by research communities. Such replicability enables research groups to easily bring up (using Archer software) local resource pools and be assured of preemptive access to their resources when needed, while providing opportunistic cycles to the community. This is in contrast to OSG and TeraGrid, which are large-scale shared physical resources not easily replicable at a small scale on local resources. Third, Archer empowers entry-level users to quickly learn HTC skills, from basic to advanced, with an interactive graphical interface hosted on their own workstations. This is in contrast to OSG and TeraGrid, where entry-level users need to learn how to operate resources that are hosted remotely, using non-interactive sessions and unfamiliar interfaces for data transfer, login, and job scheduling.

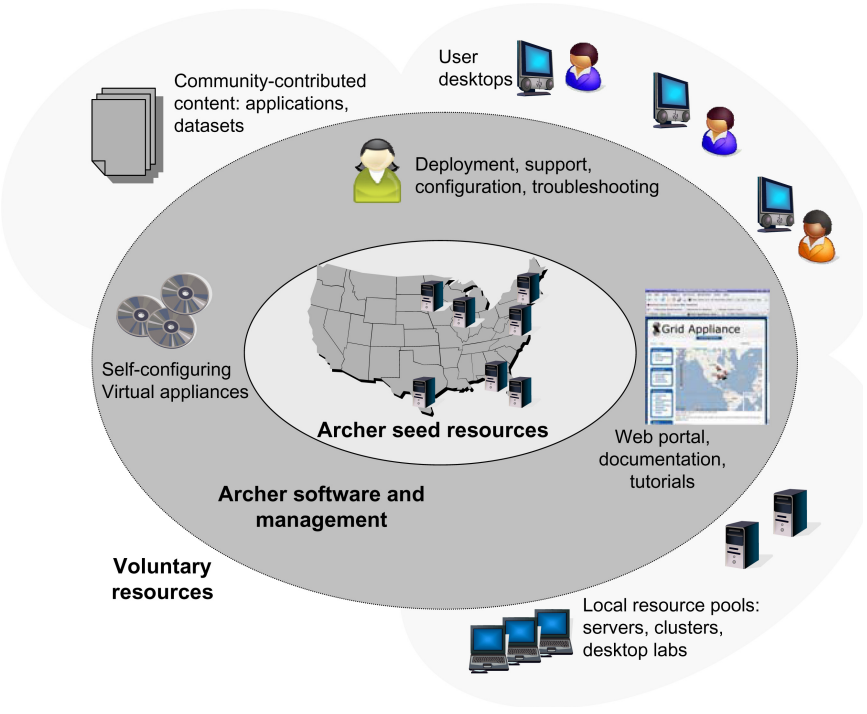

Fig. 1. Overview of Archer. The seed resources consist of seven clusters at Cornell, Florida State, Northeastern, Northwestern, U. Minnesota, U. Texas at Austin, and U. Florida. In addition to the hardware infrastructure, Archer provides virtual appliance (U. Florida) and job scheduling (U. Wisconsin) software and ready access to user-contributed data and applications. Users from non-seed sites build upon these elements to increase available resources when they join the system.

The technology in Archer provides a way to swiftly create grids of medium size, which is complementary to these projects. Figure 1 presents an overview of the Archer infrastructure.

2 Background and Motivations

In modern computer architectures, processor performance, power consumption and cost are significantly affected by design parameters and target workloads. Thus, researchers rely on simulation environments to evaluate the merit of a new idea before it is implemented in hardware. In addition, computer architects depend on high-fidelity, cycle-accurate simulation environments (including the simulators themselves and associated tools such as compilers and datasets). Because these are complex and time-consuming to develop, researchers have relied on extensible simulation environments, benchmarks and datasets developed

by others in the community — such as SimpleScalar [6], SESC [17], Simics [2], among others — as well as on modules that plug in to commercial systems, such as GEMS for Simics. The broad needs of computer architecture researchers to access high-performance resources and share simulation environments are addressed in an integrated manner by Archer. We believe that the availability of Archer encourages collaboration among groups by greatly simplifying the dissemination of applications, and increases the competitiveness of smaller research groups by providing seamless access to hardware resources and software environments. To illustrate use cases and the unique capabilities enabled by Archer, consider the scenarios described in Table 1 and illustrated by the following three fictional examples:

- *Scenario 1: High-throughput cycles for research*: Graduate student Maria at Florida State U. is preparing a paper on a novel cache design for submission to a conference. She has developed a simulator which models her design. Each simulation takes on average 12 hours to complete on her desktop, and she wishes to analyze 10 configurations on 16 SPEC CPU benchmarks. The time to run this experiment on her desktop is prohibitively large (80 days). She downloads, instantiates an Archer appliance, copies her Linux simulator binary to the VM, prepares a Condor job file (building on a tutorial), and queues 160 jobs. Archer resources are utilized at 75% capacity by other jobs; still, her simulations are expected to finish within a day.
- *Scenario 2: Local resource pooling and community sharing*: A group at Northeastern University has a local set of resources, time-shared and scheduled by students via scripts. Because the scripts do not provide load balancing, often resources become contended. They try out the Archer VM appliance and decide to join. Interacting with Archer management, they set up a local Condor pool. Their resources are load-balanced, and when not in use, they become available to other Archer users through Condor flocking.
- *Scenario 3: Collaborative development and dissemination of tools and experiments*: A joint project between Cornell and Northwestern entails the development of an environment with extensions to the SESC simulator. Graduate students Carol at Cornell and John at NWU begin development by downloading code from the SESC software repository onto Archer appliances. Carol implements and tests new features in the simulator within her VM, creates Condor scripts that vary a parameter of interest, and places her code and scripts in a shared repository. John uses Carol's code to perform experiments of his own. After several iterations, they gather data for their experiments and publish a paper highlighting their findings. They make the source code snapshot, benchmarks, and Condor scripts available on the Archer Wiki as well as on an Archer NFS (Network File System) repository which is accessible to the entire virtual network, enabling others to repeat and build upon their experiment.

Table 1. Scenarios in which users with different levels of expertise can use Archer

User	Scenario	Resources/interfaces used
Novice	Casual/trial usage of the system e.g. homework assignments in undergraduate classes	Access pre-built tools, tutorials, educational modules through interactive Web portal. No local software required but Web browser
Entry level	Small-scale experiments on Archer resources; graduate class projects	Baseline Archer appliance installed on personal workstation. Reuse tools, datasets and scripts, and pre-packaged tutorials.
Advanced	Graduate research; run medium to large-scale experiments on Archer Develop/modify simulation tools	User builds simulation tools and scripts of their own. Software installation time: 15-30 minutes
Research group	Use Archer software to manage local resources (e.g. desktop grids); deploy local/multi-site Archer pools with high priority for users belonging to research group.	Customized Archer appliance installed on PCs of researchers, lab PCs, servers and clusters. Customization and installation times: hours to days.

3 Archer Infrastructure

3.1 Overall Design Approach

The key motivations for the Archer infrastructure to be a distributed system are scalability, sustainability and dependability: new resources that join increase system capacity, the infrastructure is sustained by the community and does not overburden a single site with hosting, and the system can withstand hardware/software failures in individual sites. A distributed system, however, poses challenges in management which need to be addressed. Our system design builds on virtualization and autonomic computing techniques that specifically target ease of management. They make it possible to have effective centralized management of decentralized resources, similarly to successful infrastructures such as PlanetLab [15]. The Archer middleware integrates easy-to-install, self-configuring virtual machine appliances with virtual networks to create scalable community pools of virtual resources. Each Archer resource is a virtual appliance that is preconfigured with an installation of a Linux O/S and distributed computing middleware (Condor [16]). Archer virtual appliances are interconnected by the IPOP [9] self-configuring virtual network overlay. The choice of virtual appliances, virtual networks and Condor is motivated by the following reasons:

1. *Ease of deployment*: Virtual appliances can be easily deployed on typical x86-based machines regardless of their existing hardware/software configuration. Today's VM technologies are mature and several free virtualization options exist for Windows, Linux and MacOS systems (including VMware Player/Server, KVM, VirtualBox and Xen). Experiments with our prototype environment show that Archer virtual appliances can be deployed typically within 30 minutes by entry-level users.

2. *Software compatibility*: Virtual appliances can run unmodified, binary software, including a wealth of existing computer architecture simulators and support tools. Representative examples include SESC, SimpleScalar, PTLsim and Simics.
3. *Seamless connectivity*: The IPOP virtual network overlay which runs on Archer appliances provides bidirectional IP connectivity across all appliances. The virtual network supports nodes behind firewalls and network address translators (NATs) typically found in educational institutions and Internet service providers. The virtual network is self-organizing and packaged with the virtual appliance in a way that does not require any configuration from end users.
4. *Scalable and robust job scheduling*: Condor is a robust job scheduler used in thousands of resources across the world. It supports both unmodified binaries and Condor-linked applications, facilitates the queuing and management of large numbers of jobs, and has been successfully demonstrated to be effective in supporting a variety of computer architecture simulation workloads.
5. *Isolation*: Virtual appliances are isolated from their hosts. Undesirable behavior is confined to a VM, which can be easily shut down and restarted from scratch by its user.

3.2 Archer Core Middleware

Virtual Machines

Archer builds on the Grid appliance system [21], which is a self-configuring virtual machine appliance that packages, in an easy-to-deploy image: a Linux distribution trimmed for size, the Condor high-throughput computing scheduler [16], the IPOP [9] virtual network, and system management scripts that automate the process of joining the Archer virtual network. Figures 2 and 3 show screenshots of the Grid appliance.

Classic system VMs were originally developed to enable efficient time-sharing of mainframe computers by multiple independent applications and O/Ss [12,19]. They are implemented by means of VM monitors (also known as hypervisors), which are responsible for intercepting and emulating the execution of privileged instructions that deal with shared resources: CPU, memory and I/O. VM technologies have evolved quite rapidly in recent years [8]. VMs now can achieve performance on par with non-virtualized systems [5], and are increasingly pervasive in commodity systems; virtualization extensions are shipped with all Intel and AMD x86 processors, virtualization software is available from a variety of vendors (VMware, Microsoft, Parallels) and in the open source realm (Xen, KVM, which has already been integrated with the Linux kernel, and VirtualBox). The isolation and decoupling properties of VMs are particularly attractive in distributed systems [7]. Virtual machines assist in the deployment of compute nodes because of their decoupling from the operating system running on the physical machine. VMs offer unique opportunities for load balancing and fault tolerance that build upon growing support for checkpointing and live migration of running VMs. Furthermore, the ability to package VM software in

easy-to-deploy virtual appliances [18] is attractive as a means to disseminate (and maintain) complex, preconfigured software and middleware stacks.

Virtual Networks

Archer VMs are decoupled not only from the physical hosts by means of the VM monitor, but also from the physical network by means of tunneling. Once instantiated, an Archer VM appliance is able to self-configure and maintain connections to other appliances via IPOP tunnels. The resulting system is akin in functionality to a Network of Workstations (NOW [4]); we term it a Wide-area Overlay of virtual Workstations (WOW [10]) because both compute nodes and network links are virtualized, and resources are distributed across wide-area domains.

Complementary to VMs, virtual networking enables isolated multiplexing of private networks providing the TCP/IP environment for communication among participating nodes. Network virtualization techniques for distributed grid computing [20,9,10,13] have been shown to provide applications their native network environments, despite the idiosyncrasies of the real physical network — in particular, the increasing use of Network Address Translation (NAT) and IP firewalls, recognized as a hindrance to programming and deploying distributed computing applications, does not impede the use of virtual network-based systems. Central to the scalability of Archer are peer-to-peer techniques for resource discovery, virtual network routing, and NAT traversal, which are described in detail in [10].

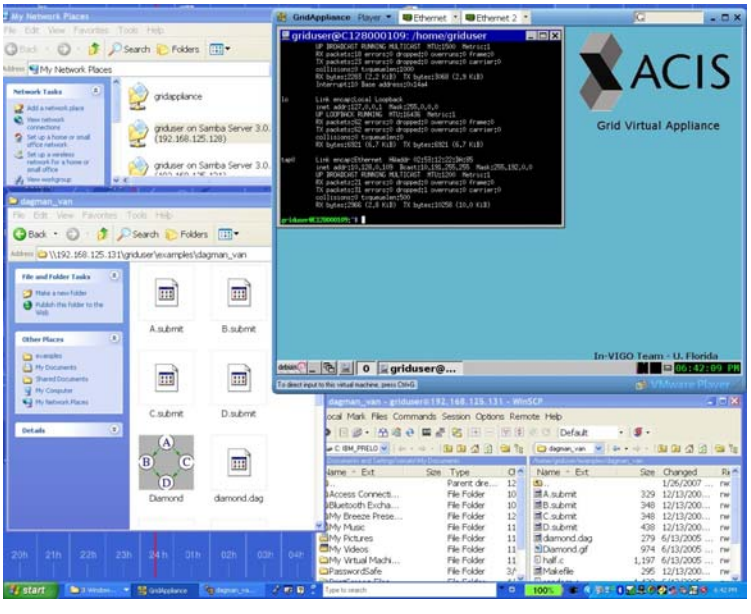

Fig. 2. Screenshot of Grid appliance in a Windows host. Top right: Grid appliance X11 user interface; top and bottom left: loop-back Samba file share allows simple browsing and drag-and-drop movement of files from host to appliance; bottom right: loop-back SSH server allows login and file transfers to the appliance.

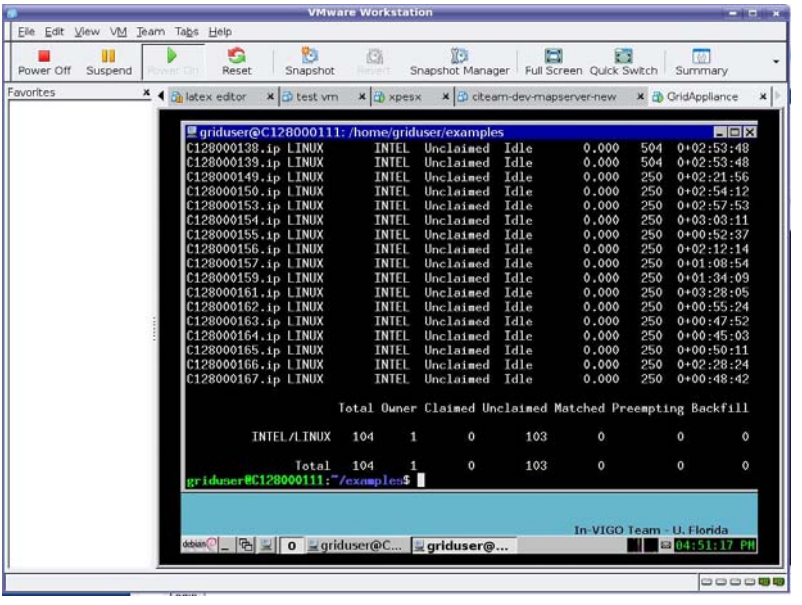

Fig. 3. The screenshot shows the Grid appliance graphical user interface and the output of the condor_status command

Condor

Condor provides easy access to large amounts of dependable and reliable computational power over prolonged periods of time by effectively harnessing all available resources, including both dedicated compute clusters and non-dedicated machines under the control of interactive users or autonomous batch systems.

Condor is an established distributed computing environment appropriate for building an ad-hoc high-throughput computing grid like Archer. Throughout this time, Condor has evolved from a local batch management system into a full-fledged distributed computing environment capable of supporting wide-area grids, complex workflows, compute-intensive applications, and data placement reliably, scalably, and with fault tolerance. It has facilities for resource monitoring, job scheduling, and workflow supervision. Current statistics show that Condor has been deployed on well more than 100,000 computers in well more than 1400 Condor pools. An important feature of the Archer system is that it creates an incentive for sites to join the infrastructure with several nodes. The pre-packaged Archer VMs provide an easy way to set up local Condor pools to manage jobs submitted by local users of a site, which are guaranteed to gain high-priority access to their resources and access to external Archer resources through flocking, while making their resources available to remote Archer users when they are idle. This kind of deployment with multiple shared pools where local control and priority is retained by individual groups has been an important feature of the GLOW infrastructure at Wisconsin, and we expect it to create further incentives for the growth of Archer.

3.3 Collaboration and Sharing

Collaboration in Archer is enabled by both the distributed virtualized infrastructure and a centralized Web-based server. The Archer Web presence includes a content management system and Web portal, with a user registration system, a Wiki (where Archer users can share documentation on how to use simulation tools in the system), and user discussion groups.

In addition to the Web-based framework for collaboration, the wide-area network of Archer virtual machines provides a basis for the deployment of file-system based data sharing frameworks that are widely used in systems, such as the Network File System (NFS). In the computer architecture field, it is desirable to use experimental data sets that include checkpointed disk and memory images containing representative benchmark workloads to drive execution-driven simulators. These disk/memory images can be large (of the order of GBytes each), as they encapsulate entire operating systems and application benchmarks. Nonetheless, in many instances these images are sparse and only a fraction of the data (few MBytes) is referenced during a simulation. Archer provides a framework for on-demand, block-level data transfer that builds on the Network File System (NFS) de-facto distributed file system implementation. Typical NFS implementations target local-area environments; Archer provides a tunneling framework that allows the deployment of wide-area NFS file systems over the encrypted tunnels enabled by IPOP. The current NFS-based deployment provides users with an automatically configured "export" folder in their appliance, where they can copy datasets that are then available via on-demand, auto-mounted read only NFS mounts.

3.4 Deployment Modes

Recognizing that the the infrastructure is used in different ways by different kinds of users, Archer supports three main deployment modes: Express, Global and Local.

Archer Express: The main purpose of Archer Express is to make it as simple as possible for a user with little background on VMs, Linux, or Condor to get started with a hands-on experiment with Archer in less than 30 minutes. Archer Express includes a small-scale public resource test pool, and a self-configuring VM appliance that does not require registration and authentication with the database of registered users.

Archer Global: The main purpose of Archer Global is to serve as the production, large-scale resource pool for the computer architecture community at large. Unlike Archer Express, where the key goals are quick deployment and ease of use, Archer Global strives to achieve scalability with strong security guarantees, by creating a private pool connecting resources of registered Archer users.

Archer Local: The main purpose of Archer Local is to allow users who wish to deploy a private Archer pool — i.e. one that runs on their own resources and

is fully decoupled from Archer Global and Archer Express. With Archer Local, users are in control of the entire configuration and management of their local cluster, independently from other Archer resources.

3.5 Security Considerations

By utilizing VMs, virtual networks and Condor, Archer provides several levels of isolation among users and with respect to the physical infrastructure. The only access that external users have to any Archer VM is through Condor, as an unprivileged user "nobody" — no direct logins are allowed. The VM runs only essential middleware services to minimize the possibility of privilege escalation within the VM. Even if privilege escalation does happen, users are confined to a virtual machine sandbox and do not have direct access to the underlying physical resources. The TCP/IP traffic that is generated by a VM is completely confined to the virtual network, as described in [21]. Archer hosts are authenticated and traffic is encrypted end-to-end by deploying a security stack in each VM based on public key cryptography (PKI). In other words, Archer VMs are only able to communicate with other Archer VMs, preventing the use of Archer resources to initiate denial of service or other kinds of attacks to physical resources. There are a couple of exceptions to this rule, which are necessary for Archer VMs to be accessible from physical resources so users can interact with them. We establish communication channels between Archer VMs and physical hosts using host-only virtual networks (software-emulated networks confined to a single host) that are carefully controlled to provide only two types of services: secure shell logins, and access to user data within the VM through Samba and NFS file systems. In a typical usage case of a Windows-based desktop, a user deploys an Archer VM on their desktop, interacts with the X11-based graphical user interface in the VM through its console, logs into the VM from the physical host using SSH, and browses a Samba network share exported by the appliance (accessible only within the host) to copy data to and from the VM. Security patches are regularly applied to the baseline Archer VMs and made available for upgrades; the process of upgrading VM appliances is facilitated by the use of UnionFS stacked file systems, whereby it is possible for users to upgrade the base system configuration of the appliance by simply replacing a virtual disk file and rebooting the system. User data and local configurations remain unmodified in the VM after the upgrade as they are stored in different stacks of the file system.

3.6 Performance Considerations

The advantages in isolation, security and management provided by VMs connected over virtual networks have associated performance overheads due to the VM and virtual network tunneling. However, these overheads are often small for CPU-intensive applications which are typical in computer architecture simulation. Studies have quantified this overhead under different scenarios, showing that the overhead CPU-intensive applications such as SPEC benchmarks is a few percent points [5], and in [21] it has been shown that the virtual/physical

overhead for a Xen VM connected to a virtual network approximately 1% for a 37-minute Condor-scheduled SimpleScalar run (sim-cache, "go" benchmark). The following experiment illustrates the capabilities and performance of the middleware and software of Archer in the context of computer architecture research and education. We have run a simulation experiment in our prototype Archer deployment, in which 200 jobs were submitted from a virtual appliance. Each job consisted of a cache simulation running 1 billion instructions of the SPEC benchmark "go" for different cache configurations arising from varying level-1 and level-2 cache sizes and associativities. The jobs were submitted from a laptop running the Archer virtual appliance behind a broadband (1MB/s) network provider. The virtual appliances in which the jobs executed were distributed across five universities (including U. Florida, U. Minnesota, Northwestern University), making up a pool of 56 VMs.

The total execution time to finish all 200 jobs was approximately 7.5 hours. If these jobs were to be executed on a single node, the execution time would have been 9.5 days assuming the median single job times measured across the 56 heterogeneous resources. Figure 4 shows a plot of the cumulative distribution of number of jobs completed as a function of time. The virtualization overhead for this application using VMware-based VMs was measured to be 11 percent, which is acceptable given the goal of achieving high throughput. Reducing the overhead introduced by

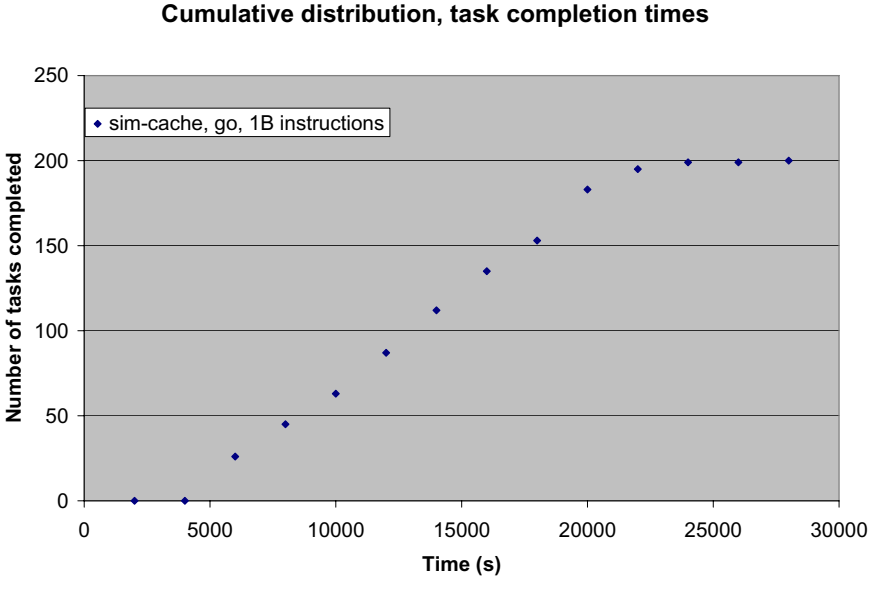

Fig. 4. Distribution of simulation completion times for an experiment with 200 SimpleScalar sim-cache jobs executed on a 56-node prototype Archer system. The median and average single job execution times are 4080 and 4320 seconds, respectively. In steady state the system was completing jobs at an approximate rate of one job every 90 seconds, compared to the throughput of one job per 42 minutes of a single job running on a single resource.

virtual machines is an active area of research and development in academia and industry, and the expected trend is for these overheads to be reduced.

4 Related Work

Inspired by projects which have been extremely successful at bringing large number of voluntary resources, such as SETIhome [3] and other BOINC-based (Berkeley Open Infrastructure for Network Computing) projects, Archer also allows nodes to join the infrastructure seamlessly. The key difference in Archer is that, in BOINC-based systems, applications need to be modified to use their application programming interfaces, and users are constrained to donating resources only. In contrast, in the Archer infrastructure, the computing node sandboxes are system VMs capable of running existing, unmodified binary applications, which is critical for adoption by the computer architecture community. Furthermore, users are able to both donate and make use of Archer resources through their virtual appliances.

Archer is closely related to PlanetLab [15] with respect to how resources are distributed and managed. PlanetLab is also a distributed system where individual researchers across many sites contribute to the overall aggregate capacity of the system by providing locally managed physical hardware (805 nodes at 402 sites worldwide, as of July 2007, while the middleware and software is managed in a centralized manner (by PlanetLab Central). However, Archer differs fundamentally from PlanetLab in purpose. PlanetLab is a generic testbed for experimental networking research and does not support load balancing of jobs, while Archer targets compute-intensive applications. Archer is also different in that it does not require dedicated non-firewalled physical machines.

RAMP (Research Accelerator for Multiple Processors [1]) is a related resource for the computer architecture community. The focus of RAMP is on the use of programmable logic to speed up the simulation of large-scale multiprocessors. While RAMP provides the potential for large speedups over software simulation, it requires users to develop their simulation infrastructure to match the specific RAMP software and hardware stack. Archer, in contrast, is general-purpose and supports a wealth of unmodified single- and multi-processor simulation tools that computer architecture researchers already use in their own local environments (e.g. SimpleScalar, SESC, Simics), offering a lower barrier of entry to its use. Nonetheless, Archer and RAMP are complementary resources in that they focus on different aspects of quantitative computer architecture research: general-purpose simulation in Archer, high-performance multi-processor simulation in RAMP.

Archer is related to Open Science Grid, where resources are pooled across institutions using a consistent software base packaged for ease of configuration, deployment and maintenance of middleware (Virtual Data Toolkit, VDT), and TeraGrid, a high-performance infrastructure well-suited to run large parallel jobs. Aside from the fundamental differences in goals, Archer differentiates from these systems with respect to its technology: the use of VM-based appliances

for software distribution and self-configuring virtual networking to facilitate the addition of resources to the infrastructure. Also, Archer is targeted at serving a single rather than multiple communities, which enables its content to be tailored to the interest of computer architects by the architecture community.

Archer is similar to the Intel NetBatch infrastructure in its support for high-throughput simulation workloads. NetBatch is also a distributed system consisting of CPUs distributed across multiple sites, managed by an in-house batch scheduler. It has been highly successful in providing batch computing cycles for a variety of applications at Intel: it started with hundreds of computers in 1990, and over the course of ten years grew to 10,000 nodes across 25 sites, logging 2.7 million jobs per month in their queues [11]. Archer is different from NetBatch in that it is not internal to a private corporate network, allowing individuals to easily join and contribute resources to the system.

5 Conclusions and Future Work

This paper describes a novel infrastructure that integrates virtual machine appliances interconnected by virtual networks with Grid high-throughput scheduling to create wide-area collaborative environments for sharing of CPU cycles, data sets and documentation. An initial deployment of Archer with 56 CPUs began in the Summer of 2008 and has been successfully exercised with representative open-source and commercial computer architecture simulators (SESC, SimpleScalar, and Simics). The infrastructure is being extended with three distributed clusters (112 cores and 4TB of storage each) in the Fall of 2008.

The work described in this paper focuses on the infrastructure as tailored to the needs and tools of the computer architecture community. Nonetheless, the approach taken in Archer can be applied to other areas of engineering and science. As an example, related work also uses the core virtualization in Archer (the Grid appliance) in support of simulation tools for a community of coastal and estuarine scientists (SCOOP).

Directions for future work in Archer include the ability to incorporate performance and availability enhancements on top of the baseline NFS data sharing infrastructure through user-level caching proxies, request redirection to replica servers, and resource discovery using IPOP's distributed hash table. In addition, a complementary approach for cooperative whole-file transfers using a FUSE virtual file system with BitTorrent transport is being developed.

Another area of future work involves the integration of social networking frameworks with Grid appliances, which will enable the creation of a fourth deployment mode for the infrastructure — Archer Social — where ad-hoc groups of users will be able to easily bring up resource pools by leveraging Web-based online social networking infrastructures to enable simple creation of social virtual private networks and clusters [14]. We have a prototype of the Grid appliance integrated with Facebook APIs aimed at facilitating the creation and management of local and cross-domain private pools dedicated to groups of collaborators.

Acknowledgments

This work is sponsored by the National Science Foundation under CRI collaborative awards 0751112, 0750847, 0750851, 0750852, 0750860, 0750868, 0750884, and 0751091. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Ramp web site, `http://ramp.eecs.berkeley.edu` (accessed, October 2008)
2. Virtutech simics web site, `http://www.virtutech.com` (accessed, October 2008)
3. Anderson, D., Cobb, J., Korpella, E., Lebofsky, M., Werthimer, D.: Seti@home: An experiment in public-resource computing. Communications of the ACM 11(45), 56–61 (2002)
4. Anderson, T., Culler, D., Patterson, D.: A case for network of workstations: Now. IEEE Micro (February 1995)
5. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the nineteenth ACM symposium on Operating systems principles, Bolton Landing, NY, pp. 164–177 (2003)
6. Burger, D., Austin, T., Bennett, S.: Evaluating future microprocessors - the simplescalar toolset. Technical Report 1308, Computer Science Dept. University of Wisconsin (July 1996)
7. Figueiredo, R., Dinda, P., Fortes, J.: A case for grid computing on virtual machines. In: Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS), Providence, Rhode Island (May 2003)
8. Figueiredo, R., Dinda, P., Fortes, J.: Resource virtualization renaissance. Guest Editor's Introductionm, in special issue on Virtualization 38(5), 28–31 (2005)
9. Ganguly, A., Agrawal, A., Boykin, P.O., Figueiredo, R.: IP over P2P: Enabling self-configuring virtual IP networks for grid computing. In: Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rhodes, Greece (June 2006)
10. Ganguly, A., Agrawal, A., Boykin, P.O., Figueiredo, R.: WOW: Self-organizing wide-area overlay networks of virtual workstations. In: Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC), Paris, France (July 2006)
11. Gelsinger, P.: Keynote speech, intel developers forum, Fall (August 2000), `http://www.intel.com/pressroom/archive/speeches/pg082400.htm` (accessed November 2008)
12. Goldberg, R.: Survey of virtual machine research. IEEE Computer Magazine 7(6), 34–45 (1974)
13. Jiang, X., Xu, D.: Violin: Virtual internetworking on overlay infrastructure. In: Cao, J., Yang, L.T., Guo, M., Lau, F. (eds.) ISPA 2004. LNCS, vol. 3358, pp. 937–946. Springer, Heidelberg (2004)
14. St Juste, P., Wolinsky, D., Xu, J., Covington, M., Figueiredo, R.: On the use of social networking groups for automatic configuration of virtual grid environments. In: Proceedings of Grid Computing Environments (GCE) Workshop, Austin, TX (November 2008)

15. Culler, D., Peterson, L., Anderson, T., Roscoe, T.: A blueprint for introducing disruptive technology into the internet. In: Proceedings of HotNets-I 2002 (October 2002)
16. Litzkow, M., Livny, M., Mutka, M.: Condor - a hunter of idle workstations. In: Proc. 8th IEEE International Conference on Distributed Computing Systems (ICDCS) (June 1988)
17. Renau, J., Fraguela, B., Tuck, J., Liu, W., Prvulovic, M., Ceze, L., Sarangi, S., Sack, P., Strauss, K., Montesinos, P.: Sesc simulator web site, `http://sesc.sourceforge.net` (accessed, October 2008)
18. Sapuntzakis, C., Lam, M.: Virtual appliances in the collective: A road to hassle-free computing. In: Proceedings of 9th Hot Topics in Operating Systems (HotOS) (May 2003)
19. Smith, J., Nair, R.: Virtual Machines: Versatile Platforms for Systems and Processes. Morgan Kaufmann, San Francisco (2005)
20. Tsugawa, M., Fortes, J.A.B.: A virtual network (vine) architecture for grid computing. In: Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rhodes, Greece (June 2006)
21. Wolinsky, D., Agrawal, A., Boykin, P.O., Davis, J., Ganguly, A., Paramygin, V., Sheng, P., Figueiredo, R.: On the design of virtual machine sandboxes for distributed computing in wide-area overlay of virtual workstations. In: First IEEE Workshop on Virtualization Technologies in Distributed Computing (VTDC) (2006)