Preliminary version. Final version appears in First Workshop on Automated Control for Datacenters and Clouds (ACDC09). In conjunction with the 6th Intl. Conference on Autonomic Computing and Communications (ICAC 2009). Barcelona, Spain. June 19, 2009.

Applicability of the Willow Architecture for Cloud Management

Zach Hill, Marty Humphrey
E-Science Group, Department of Computer Science, University of Virginia

Abstract. *The differences between Grids and Clouds arguably include objectives, organization, scale, and workload. By examining these fundamental characteristics and requirements in detail, we assess the degree to which a generic management infrastructure for Grid computing can be applied to Cloud infrastructures. Our analysis is further refined by considering a specific management system, Willow, which we have recently successfully applied to Grid management. Three distinct architectures are evaluated: wholly within the datacenter transparent to users; completely within the user software stack and transparent to the Cloud provider; and, a hybrid in which the system is known to both the provider and the users.*

1. Introduction

As "Cloud" computing, the union of utility computing and Software-as-a-Service [1], becomes more prolific, the datacenters that support it will become both larger and more numerous. In these datacenters, the ratio of servers to operations staff member must continue to increase to keep costs reasonable and minimize human error. At the same time, developers are adopting the Cloud model at an increasing rate and are demanding more guarantees in terms of quality of service (QoS). Current Cloud offerings give very few service guarantees (e.g., Slicehost's assertion that "most hosting SLA agreements are just plain silly" [2]), which places additional burden on service developers who must build robust services on-top of potentially undependable infrastructure. This either requires additional complexity, such as self-management capabilities, in the services themselves or increased guarantees and tools by the infrastructure provider.

Generally, the datacenter challenge is to extract high utilization and dependability from the resource set while meeting all service level agreements with customers. This is the case regardless of whether the platform is low-level, such as Amazon's EC2 [3], mid-level, such as Microsoft's Windows Azure [4], or high-level, such as Google's AppEngine [5]. Further, while we are beginning to see open-source Cloud computing tools such as Eucalyptus [6] and Enomalism [7] that enable advanced academic Cloud research, there is very little public information about how industry Cloud providers manage their infrastructure and what the level of sophistication is because their solutions are proprietary.

We assert that high-performance computing and Grid technologies specifically have been in production for some time and present many similar management challenges to those of the Cloud model. Both are built on large-scale shared infrastructures that must be multiplexed between many users simultaneously and which may be widely distributed geographically. This naturally raises the question: what lessons from managing Grids can be applied to management of Clouds? How do the requirements/properties of Grids compare to those of Cloud infrastructures and how do the similarities and differences influence management requirements and strategies?

In this paper we assess the degree to which a generic management infrastructure for Grid computing can be applied to Cloud infrastructures. Our analysis and discussion is further refined by considering a specific management system, Willow [8], which we have recently successfully applied to Grid management. We design and analyze three separate architectures for Cloud management based on Willow. Each architecture has different benefits/limitations and is applicable for different use-cases. Willow is capable of management at many levels in the service stack thus permitting the same basic management infrastructure to handle hardware management, virtualization management, and application/service level management simultaneously.

2. Extending the Willow Survivability Architecture for Grid Management

The Willow architecture was originally designed to ensure survivability for large-scale distributed systems. As shown in Figure 1 for Grid management, Willow consists of a hierarchy of control loops that sense and analyze the state of the system (Spartan [9]), and a powerful management mechanism to effect the required changes on the underlying system (ANDREA [10][11]).

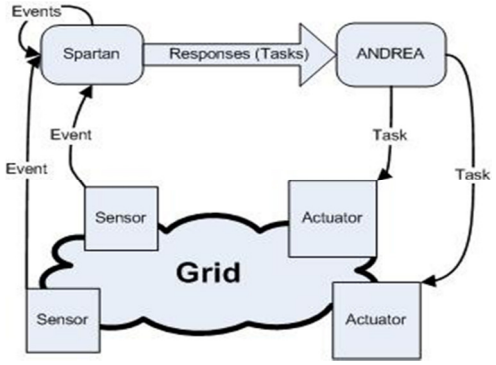

Figure 1. Willow Control Structure

It is the hierarchy of finite state machines in Spartan that allows Willow to provide a wide range of different control capabilities. At each level the sensor data is analyzed and decisions are made relative to the local environment of each node. Events (timestamped name-value pairs) are transmitted to nodes higher in the hierarchy thus giving

higher level nodes a more abstract view of the system. Thus, for example, the top-level node (finite-state machine) in the hierarchy will be concerned with system-wide issues, the next level with issues faced by regions of the system, etc. It is the top-level node, in particular, that can make determinations on courses of action without having to know who, where, or how many nodes there are in the system.

The control policy is written in the Time-based Event Detection Language (TEDL) [9]. A set in TEDL is a collection of events whose time stamps fall within some defined range relative to the local time. Predicates are then defined relative to attributes of sets. The change in value of a predicate during operation is the source of action and higher-level event generation. As the machine transitions between states based on input events, actions are triggered, which may be either to output an event to another finite state machine or to create a task and run that task on the system in order to alter the state of the system. Management action conflicts that can arise between different levels in the Spartan system are resolved through prioritization. Thus, the management system can be constructed to ensure that Grid-wide management changes take precedence over local changes or vice-versa, but consistent local changes will still be applied.

Once a Spartan node determines that a task should be executed, it creates it and passes it off to ANDREA to see that it is executed on the nodes of the monitored system. Data about the effect of the task is gathered by Spartan from sensor events it receives after the task is executed. ANDREA is implemented as a publish-subscribe overlay network. The network passes tasks based on subscription attributes stating which sources and types of tasks a node is willing to accept. A task in ANDREA is a *workflow* that features most typical workflow constructs such as delegation, sub-tasks, and conditional tasks (if-then-else). In effect, addressing nodes within the system is based on attributes that the nodes have and not on specific network addresses or some other form of identification. Examples include the name of a single node, or can be a general property of a node, e.g. any machine running Fedora Core 6. Tasks in ANDREA are considered to be constraints on the configuration of a system. Enacting a task on the Grid therefore equates to constraining the configuration of the Grid to conform to some requirement. ANDREA holds the constraint until it is told to do otherwise.

Figure 2 shows the architecture of experiments we performed to assess the feasibility of Willow for some specific problematic Grid use-cases (see [12] for details). Because we could not acquire test resources *at scale*, the scenario we implemented was relatively modest but sufficiently organizationally complex to represent realistic environments such as the NSF/DOE Open Science Grid [13] ("GT4/GRAM" in Figure 2 refers to Globus Toolkit v4.0.2 [14], the Grid software that we were managing). The goal of our experiments was to determine if we could sense that certain time-dependent scientific jobs were not completing in time and to dynamically reconfigure the Grid to make resources available for these important jobs. The key finding in this research was that we were able to specify relatively simple local policies/sensors/actuators (even further simplified in Figure 2) and define higher-level policies that act on the statistical aggregate of this local information to implement an efficient and effective global response. In particular, the fact that jobs were completing too late at "UVa" (which without our system would be viewed as merely a "local" event), in combination with jobs

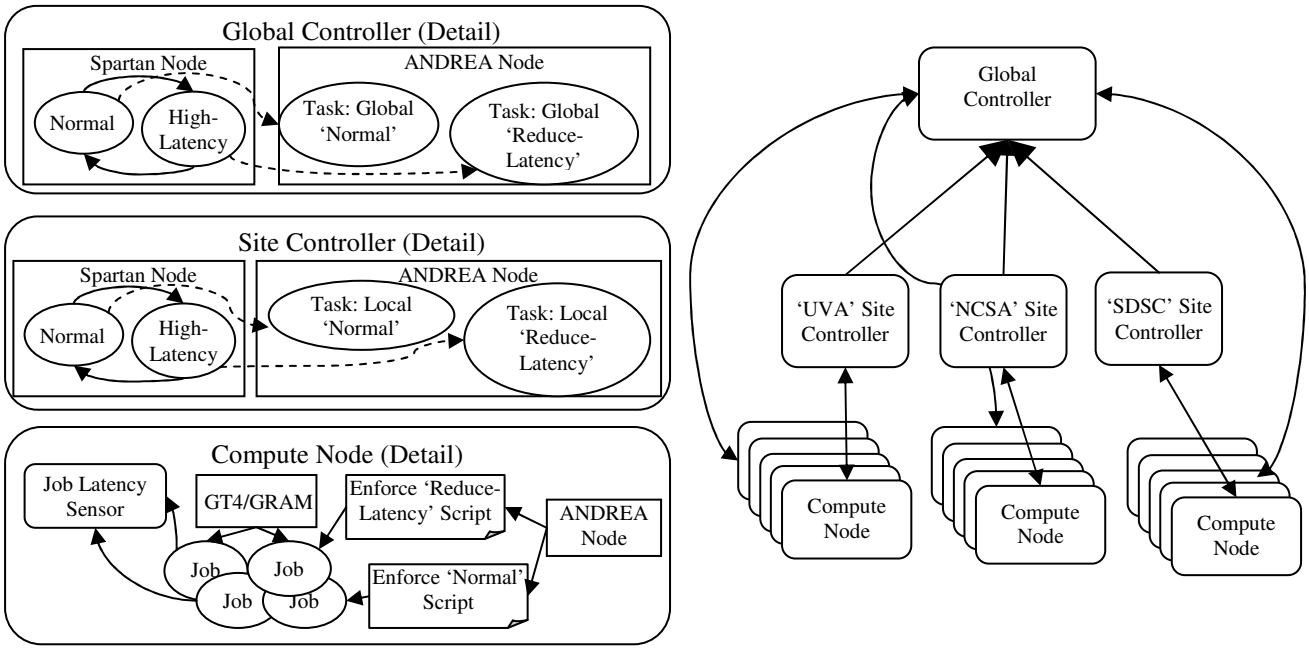

Figure 2. Adapting Willow for Grid Management (Experiment Configuration of [12])

completing too late at "NCSA", was the basis for Willow to proactively shift resource availability at "SDSC" *before* jobs at "SDSC" started to terminate after their respective deadlines. Such proactive, global-scale action enabled significantly better performance as compared to the existing, myopic, local reaction and reconfiguration. Our results show that automated management is feasible and particularly promising for the management of existing large-scale Grid deployments. Our results show that by configuring the Grid using a holistic view and utilizing site-external context, the overall performance, security, and dependability was enhanced. In this continuing research, we are enhancing the Grid management capabilities to better support site autonomy explicitly and providing a policy-based resource usage control for Grid middleware.

3. Applicability of Willow to Cloud Management

To assess the applicability of the Willow architecture for the Cloud environment, we first determine the unique features of Cloud computing as compared to Grid computing. We derive three potential architectures based on this analysis: infrastructure-level, user-level, and a hybrid of the two. We analyze the pros and cons of each approach.

3.1 Characteristics of Grids and Clouds and the Implications for Automated Management

We believe that the key differences between Grids and Clouds with regard to the potential for automated management infrastructures are objectives, organization, scale, and workload characteristics.

Objectives. It has been asserted that Grid technology's objective is to give a relatively small number of users access to a very large pool of very high-performance resources through cooperation than they could have at any individual institution [15]. Thus, the management goal is to enable efficient scheduling and allocation of resources to users and coordinate widely-distributed resources so that a single user can utilize them fully. Cloud computing on the other hand, gives a large number of users the exact amount of resources that each requires at any given time and to relieve the user of common infrastructure management duties. Further, the objective is to utilize economy-of-scale and multiplexing of resources to offer scalable infrastructure cheaply.

Organization. The typical Grid topology is hierarchical in nature with a relatively small number of clusters/resources, hundreds at most, each of which may have upwards of 1000 nodes. Because each site is individually managed by the local owners, the ratio of operations and administrative staff to servers in this model is relatively high compared to that of Cloud computing. The distributed ownership of resources in the Grid significantly compounds the difficulty of automating management across domain boundaries. Cloud infrastructures however tend to be organized much differently. The topology tends to be much flatter as nearly all nodes in the infrastructure are reachable externally. Additionally, because Cloud infrastructures are owned by a single entity the cross-domain challenges present in the Grid environment are not present. However, if a user spans his application across multiple infrastructures and multiple providers then the multi-domain constraint is present, albeit at a higher level.

Scale. While Grid systems have certainly achieved significant scale, into the 100,000s of processors, the deployment and growth of Grid has not matched that of Cloud computing in which providers are building new datacenters containing 10,000 servers or more with regularity and the total number of processors found in the Cloud far outpaces that of event the largest Grids. Scale is a first-class concern in both Clouds and Grids, but because of the organization of Cloud infrastructures and the requirement to keep operations costs low, handling scale and dynamism is even more important in Cloud management than is found in Grids.

Workload. Grid usage patterns and workloads tend to focus on two primary classes of work: tightly-coupled HPC jobs, often utilizing MPI; or, embarrassingly parallel "bag of tasks" computations which do very little inter-process communication but require lots of cycles. Cloud computing on the other hand tends to be predominantly request/response or query-based, meaning it must handle high numbers of relatively short-lived transactions or requests. Grids, particularly for the tightly-coupled long-running MPI computations, require high reliability of a large set of the nodes as a single failure may cause a long-running job to fail resulting in the waste of significant resources. Cloud infrastructure must be able to handle server failure because at datacenter scale servers will always be failing. Because Grid systems typically allocate a resource for the exclusive use of user, a failure may only impact one user, whereas in the Cloud environment resources are multiplexed and thus a single failure may impact several users. Clouds also generally have one advantage over most Grids in terms of achieving reliability and availability—the fact that the users see only virtualized resources means that the providers can utilize replication and other mechanisms to achieve high perceived availability and reliability even if the underlying hardware is not. Grids, usually being built on bare-metal for pure-performance reasons, do not have this capability.

3.2 Willow Architectures for Cloud Management

Based on our previous research in using Willow for Grids, and based on our analysis regarding the critical differences between Grids and Clouds as it pertains to potential automated management, we have designed three architectures for deploying Willow in the Cloud environment. For each architectural choice, we give scenarios that demonstrate its utility and assess its general advantages/disadvantages. We do not advocate that one

particular architecture is superior to the others, but that each has trade-offs. Our future research is to quantify each architecture's effectiveness in a broader set of situations.

3.2.1 Infrastructure-level Willow Architecture

The first approach is to use Willow entirely within the walls of the datacenter. In this organization Willow is essentially invisible to the user of the Cloud-service.

One example is deploying Willow alongside a Eucalyptus [6][15] installation. In this scenario, Willow can provide error-detection and system protection services and would monitor the actions of the various Eucalyptus control nodes themselves (the Node Controller, Cluster Controller, and Cloud Controller). In addition to monitoring the virtualization-layer management systems, Willow can also be used to monitor the networking infrastructure as well as other physical assets such as cooling or power-delivery equipment. As long as these systems have networked interfaces and a management protocol it is possible to build a software-based Willow sensor to monitor its health and/or performance and integrate that information into the state of the system as a whole. Such a deployment increases the dependability of the overall system in situations where a Eucalyptus (EPC) Cluster Controller fails. Willow, sensing the failure, could automatically switch to a hot-backup by reconfiguring the EPC Cloud Controller.

In another scenario such a deployment increases the ability of Eucalyptus to meet its Service Level Agreements (SLAs) for network bandwidth. For example, one could imagine a cluster, referred to as cluster Alpha, of servers that each have a pair of network interface cards (NIC) and are each connected to a pair of switches—see Figure 3. In the event that one of those switches were to fail it would halve the effective bandwidth between servers and between cluster Alpha and the other clusters in the Eucalyptus Cloud—assuming that both switches are connected to the external network as well. In such a case, a Willow sensor monitoring the switches would detect that one had failed and while all servers in the cluster Alpha are still reachable the bandwidth is reduced. The Willow Cluster Controller could reconfigure the EPC Cluster Controller to either no longer accept new service requests or inform the EPC Cloud Controller that it should migrate some VMs from cluster Alpha to another cluster depending on the existing load on the failing cluster.

Such an application of Willow to a Eucalyptus deployment is transparent to the users of the Cloud, but enables the Cloud to deliver increased dependability and maintain service levels even in the face of failures which would otherwise be visible to the users. The advantages of such an architecture are that it requires no adaptation or action by the users and that only minimal changes to the core resource management components are required, if any, to allow Willow to interact with the control nodes. The disadvantages of this architecture are that because it functions below the user-level it cannot make decisions that require knowledge of the user's application architecture or dependencies and that it may add complexity to the resource management components as part of making them configurable for Willow.

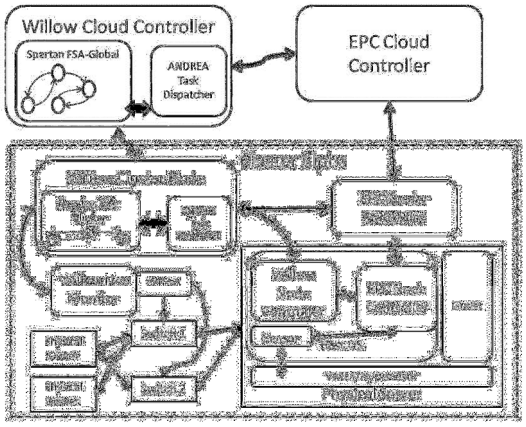

Figure 3. Willow deployed alongside the Eucalyptus architecture.

3.2.2 User-level Willow Architecture

Virtualization and Cloud computing do not guarantee that a user's service does not have bugs or that the VM image he is using has a bug-free operating system. Failure of either the user's services or guest operating systems is a real issue. As a result, it is useful for users to have a management system for their code and to help monitor their application's health and performance. Willow can fulfill this role and provide the user with multi-level monitoring and dynamic service reconfiguration and adaptation to further increase availability and performance.

In this case Willow provides a service similar to that of other third-party Cloud management products such as RightScale [16], but is extended by the user into his services directly. It might also easily be used to manage resources in the Cloud spread across multiple Cloud vendors and even mixed with local computing resources. Willow could not only be used to monitor and manage a user's virtual infrastructure, but also the applications and services hosted on the virtual infrastructure. Utilizing Willow for service reconfiguration has the additional advantage that the logic the user deploys in Willow can have higher-level knowledge of the semantics of the application itself including communication and performance requirements of the application components themselves rather than just the virtual infrastructure.

As an example, consider the case where a user has a typical 3-tiered Internet service. Willow could be configured as shown in Figure 4 and could include service-level sensors to monitor request response time, virtual machine CPU utilization, and network load on each virtual machine of each tier. The global application controller might have three states: "steady demand", "increasing demand", and "decreasing demand". If sensors report that load is

increasing above a set threshold then the controller would go into 'increasing demand' mode and request new virtual machines and redistribute load as needed. In the opposite case, where demand and load are decreasing, the controller would go into "decreasing demand" mode and release resources to minimize costs. Clearly, if no significant demand change is detected the "steady demand" state would be selected and no changes made. Using the fact that the service is broken into three tiers and the relationships between them the application controller might automatically adjust the other tiers in response to demand changes in one tier, thus maintaining proper resource ratios based on sensor input rather than pre-calculated numbers. Additionally, by monitoring both the service's response time and the virtual machine's utilization it would be possible to detect the difference between a service failure and a virtual machine crash and an appropriate response issued. Another benefit is that it would be possible to monitor the correlation between service response time and virtual machine utilization and adjust the resource pool size to achieve the desired values for either metric independent of the other. Such a capability would aid in debugging as well as scaling efficiently.

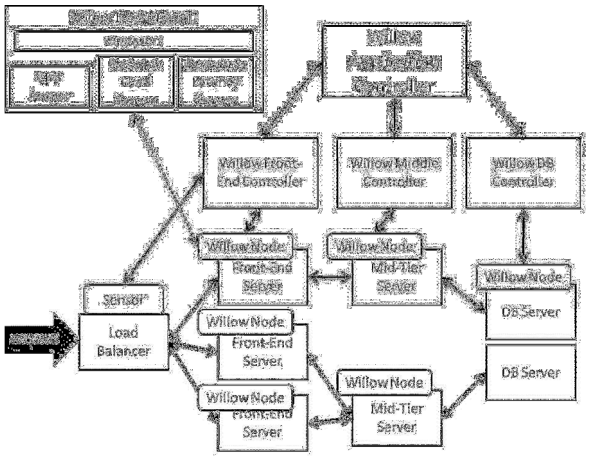

Figure 4. Willow deployed to manage a tiered web application.

The advantages of this deployment architecture are that the user is able to encode service-specific knowledge into the control system (Willow) enabling higher-level decision making and that such an architecture can span multiple Cloud infrastructures. The disadvantages of this architecture are that it may require additional complexity placed in the application components to allow run-time reconfiguration and that at least some portion of the Willow architecture itself is hosted in the cloud requiring extra resources to be leased by the user, which can increase cost.

3.2.3 Hybrid Willow Architecture

The least intuitive but potentially most exciting application of Willow to Cloud management is a hybrid approach by which a management infrastructure exists on both the provider and user sides of the infrastructure and they can interact directly, as shown in Figure 5. This approach is unique in regard to the communication between the management functions utilized by the Cloud provider and the user's own application management infrastructure. This cooperation enables the user's knowledge of the semantics and organization of his system to give hints to the Cloud infrastructure management so that it may further optimize the management of the physical resources in the datacenter. Further, it is possible that this interaction could take place across multiple Cloud providers.

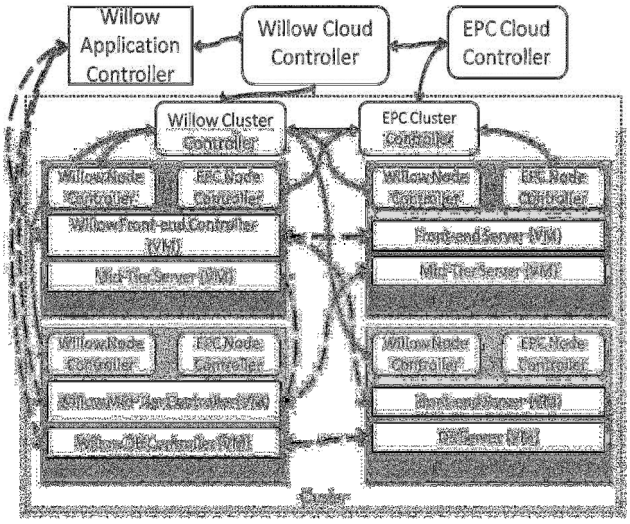

Figure 5. A hybrid approach with Willow in both the datacenter and in 'user space'.

An example of cooperation is that the user-side management system could monitor service performance and demand and inform the provider-side infrastructure management system that in the near future it expects the demand for service to increase significantly so it might be requesting additional resources soon. These enables the resource managers to perform pre-fetching of resource lists and identify resources to allocate without actually allocating them and thus the user is not charged for the resources if they are not used and the provider may instead hand the resources over to other users with more immediate requests.

Another example is that if the Cloud provider suffers a service outage where some set of resources is not longer reachable externally, the provider-side management system could inform the user-side system directly that the problem is in the provider's infrastructure rather than the user's instances having failed due to a user software bug or incorrect configuration. This would enable the user-side management system to start new instances with another provider or shift load to other infrastructures rather than trying to restart services in the unavailable infrastructure.

Defining the interfaces between the user-side and provider-side management systems and enforcing failure isolation between them is critical, but this approach seems to offer benefits which the previous two architectures cannot

and which have yet to be explored in depth. Deploying Willow in such a configuration involves two nearly disjoint deployments with a well-defined set of sensors and actuators which would cross the boundaries. APIs exposed as part of the Cloud interface could be sufficient given that enough flexibility is provided that meaningful interactions can take place. Exactly what information would cross the boundary and how each side might use that information is an area to be explored.

The advantages of this architecture are that the user can provide application specific knowledge to the underlying infrastructure's management system which can increase the efficiency of resource management. Even small amounts of information about the user's application architecture could be very beneficial to the infrastructure managers. From the user's perspective any additional information about the status of the infrastructure itself could be very useful in designing robust applications. The disadvantages are that the users interface at list somewhat with the underlying management system and that trust cannot always be placed in the accuracy of any user-provided data particularly when it comes to resource usage. Not only does it provide a technical problem but a problem of user incentives to provide accurate data. The sheer number of users all of whom could interact with the infrastructure manager also presents a challenge in that it could present so much information that the manager is overwhelmed and cannot effectively use any of it. And, while users generally prefer more information about the state of the infrastructure, this can also begin to erode one of the basic properties of Cloud computing, which is that the user shouldn't know or care where the resource actually is or what it is. By peeling back some of the abstraction and exposing any amount of detail developers could become reliant on no-guaranteed and non-specified behaviors which are subject to change.

4 Summary and Future Work

Cloud computing presents many challenges for automated management. We have presented our past work on automating Grid management using the Willow architecture and have examined to what degree it also addresses some of the challenges of managing Clouds. We presented three basic architectures for applying Willow to automating Cloud management: datacenter infrastructure management directly, user application management, and a hybrid architecture. Applying Willow in the datacenter facilitates advanced failure recovery and isolation capabilities while applying it in the user's software stack gives users increased automation over how they use the Cloud efficiently and how they can build robust application on imperfect virtualized resources. The hybrid, a combination of the two, allows both users and providers to communicate directly and assist each other in achieving better utilization, performance, and reliability.

We are continuing to prototype and evaluate the combination of Willow and Eucalyptus and what specific policies are advantageous. For applying Willow to the user-space we are identifying common classes of applications and how Willow can enhance them. We are also exploring more fully the tradeoffs of the hybrid approach and how the interaction between the users' systems and the providers' might be defined. We are also examining how much information is needed by each party to make significant improvements in application performance and dependability.

References

[1] M. Armburst, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report No UCB/EECS-2009-28. February 10, 2008.

[2] Slicehost.com Frequently Asked Questions (FAQ). http://www.slicehost.com/questions/#sla

[3] Amazon Web Services. Amazon Elastic Compute Cloud (EC2). Amazon Web Services. [Online] 2009. http://aws.amazon.com/ec2.

[4] Microsoft Corp. Windows Azure. Azure Services Platform. [Online] 2009. http://www.microsoft.com/azure/windowsazure.mspx.

[5] Google Inc. Google App Engine. Google Code. [Online] 2009. http://code.google.com/appengine/.

[6] EUCALYPTUS.[Online] http://eucalyptus.cs.ucsb.edu.

[7] Enomaly. Enomalism. Enomaly: Elastic Computing. [Online] http://www.enomaly.com.

[8] J. C. Knight, D. Heimbigner, A. Wolf, A. Carzaniga, J. Hill, P. Devanbu, and M. Gertz. The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications. Intrusion Tolerance Workshop, DSN-2002 The International Conference on Dependable Systems and Networks. 2002.

[9] Varner, P. Policy Specification for Non-Local Fault Tolerance in Large Distributed Information Systems. M.S. Thesis. s.l. : University of Virginia, May 2003.

[10] Rowanhill, J. Efficient Hierarchic Management For Reconfiguration of Networked Information Systems. International Conference on Dependable and Survivable Systems. June 2004.

[11] Survivability Management Architecture for Very Large Distributed Systems. Ph.D. Thesis, University of Virginia. July, 2004.

[12] Z. Hill, J. Rowanhill, A. Nguyen-Tuong, J. Basney, G. Wasson, J. Knight, and M. Humphrey. Meeting Virtual Organization Performance Goals Through Adaptive Grid Reconfiguration. 8th IEEE/ACM International Conference on Grid Computing. 2007.

[13] Open Science Grid. http://www.opensciencegrid.org/

[14] Globus Alliance. Globus Toolkit Homepage. [Online] http://www.globus.org/toolkit/

[15] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov. *Eucalyptus: A Technical Report on an Elastic Utility Coomputing Architecture Linking Your Programs to Useful Systems.* s.l. : UCSB Computer Science Technical Report Number 2008-10, 2008.

[16] RightScale. Web-based Cloud Computing Management Platform by RightScale. [Online] http://www.rightscale.com.