Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

TOWARDS AN AGENT-BASED SYMBIOTIC ARCHITECTURE FOR AUTONOMIC MANAGEMENT OF VIRTUALIZED DATA CENTERS

Qi Liu

IBM T. J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598, USA

Georgios K. Theodoropoulos

IBM Dublin Research Laboratory
Damastown Technology Campus
Mulhuddart, Dublin 15, Ireland

Dilma Da Silva

IBM T. J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598, USA

Elvis S. Liu

School of Computer Science and Informatics
University College Dublin; and
IBM Dublin Research Laboratory
Mulhuddart, Dublin 15, Ireland

ABSTRACT

The increasing scale and complexity of virtualized data centers pose significant challenges to system management software stacks, which still rely on special-purpose controllers to optimize the operation of cloud infrastructures. Autonomic computing allows complex systems to assume much of their own management, achieving self-configuration, self-optimization, self-healing, and self-protection without external intervention. This paper proposes an agent-based architecture for autonomic cloud management, where resources and virtual machines are associated with *worker agents* that monitor changes in their local environments, interact with each other, make their own decisions, and take adaptive actions supervised by a *network of management processes*. To fulfill global objectives, the management processes conduct what-if simulations and update the worker agents' local rules when necessary. Such a guided decentralized decision making method can mitigate the pressure on the system management stack, improve the effectiveness of resource management, and accelerate the response to failures and attacks.

1 INTRODUCTION

Cloud computing has become an appealing technology to enable IT enterprise applications, using virtualization techniques to allocate computational resources in a pay-as-you-go manner (Armbrust et al. 2009). From a high-level view, a cloud infrastructure may be built from several warehouse-scale data centers. Within a data center, the physical resources (e.g., processors, memory, network, and storage) of each server are managed by a virtualization layer that carves the shared resources into virtual machines (VMs), which can be provisioned as needed dynamically. Thanks to the decoupling of the VMs from the underlying physical infrastructure, one can adjust the resources of the VMs to adapt to changing workload demands and consolidate workloads across physical server boundaries. Data center virtualization is the cornerstone that underlies the cloud computing paradigm, yielding such benefits as elastic scaling, high availability, increased server utilization, reduced power consumption, and disaster recovery support.

Depending on the service model presented to customers, cloud computing offerings fall into three broad categories, namely *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), and *Software as a Service* (SaaS). Among these service models, IaaS clouds, as exemplified by Amazon EC2 (Amazon 2012a) and IBM Smart Cloud Enterprise (IBM 2012a), provide the greatest flexibility to customers, giv-

ing them the illusion of having dedicated servers to host their applications. On the other hand, IaaS cloud providers face significant challenges to maximize the benefits associated with data center virtualization, partly due to the lack of application specific information at the infrastructure level. To cope with these challenges, data centers usually use sophisticated virtual infrastructure management software with layers of modules to carry out a variety of services, including admission control, resource allocation, VM life cycle management, scheduling, load balancing, utilization and power optimization, monitoring, billing, quality-of-service guarantee, intrusion detection, and failure/attack recovery (Sotomayor et al. 2009; Zhang and Zhou 2009). The interaction between these services is emergent in nature and commonly results in conflicting objectives among different services. In addition, many services require near real-time decision making in order to respond to unexpected events (e.g., network congestion, workload surge, or hardware failure) or to take proactive actions before the occurrence of adverse events (e.g., malicious attacks or security vulnerabilities). Nonetheless, most of the current management software stacks still use special-purpose controllers organized in hierarchical structures to implement these services. With the increasing scale of data centers, the controller hierarchies tend to grow vertically, leading to prolonged response time when multiple controllers in the hierarchies are involved in the decision making process. Moreover, as the heterogeneity of the underlying virtual infrastructure continues to rise, the logic of the controllers and their interactions become more complex, resulting in higher possibility of software failure and making it more difficult to diagnose runtime errors.

Envisioned as a solution for self-management of complex systems, autonomic computing uses various self-governing components to achieve self-configuration, self-optimization, self-healing, and self-protection without human intervention (Huebscher and McCann 2008). Since self-management also resembles the key characteristic of *agent-based* and *serviced-oriented* systems, it is natural to combine these technologies coherently in system design (Brazier et al. 2009). At the same time, Dynamic Data Driven Applications Systems (DDDAS/info-symbiotic) are known as particularly suitable for implementing autonomic systems (Darema 2005; Darema et al. 2010). The success of autonomic computing, however, hinges on our ability to resolve issues such as behavioral modeling, collaborative optimization, and multilateral interaction in a multi-agent dynamically adaptive environment (Kephart and Chess 2003). As noted by Kephart and Chess, *behavioral modeling* lies at the heart of these issues, because it is necessary not only to understand what global behavior would be obtained from the local behavior of individual agents, but also to derive a set of local rules that, when implemented by the individual agents, can lead to the fulfillment of desired global objectives. In other words, a *bidirectional* modeling method is required to help explore the local-global behavioral relationship in either way.

This paper proposes a multi-agent info-symbiotic architecture for autonomic management of virtualized data centers, where resource management can be done at the relatively coarse granularity of VMs instead of application processes. In this architecture, *worker agents* are attached to the physical resources and virtual machines in a data center, continuously monitoring events of interest in their local environments, interacting with other worker agents in a peer-to-peer manner, making resource management decisions according to their local rules, and coordinating their actions reactively and proactively under the supervision of *a network of management processes*. The management processes, on the contrary, are responsible for ensuring that the emergent overall behavior of worker agents can lead to the fulfillment of prescribed global objectives, while influencing the worker agents indirectly by updating their local rules asynchronously when necessary. To this end, the management processes analyze the data collected from the worker agents in parallel, maintain an up-to-date world view of the worker agents' operations, and conduct distributed what-if simulations to evaluate alternative options under different scenarios. By separating the lightweight rule execution of the worker agents from the resource-intensive data analysis and simulation of the management processes, this approach can integrate efficient online decision making with continuous background optimization in a unified architecture. Furthermore, the results of the what-if simulations can be transformed into reusable knowledge to help accelerate proactive actions later on. It is our contention that such a guided decentralized decision making method could achieve the goal of adaptive bidirectional modeling and improve the effectiveness of virtual infrastructure management.

In the rest of the paper, Section 2 reviews related work. Section 3 introduces two example open-source cloud management stacks. The agent-based symbiotic architecture is proposed in Section 4, followed by a discussion of the bidirectional modeling and simulation in Section 5. Section 6 presents a use case study in data center business resilience assurance, while Section 7 outlines potential implementation challenges. Section 8 concludes the paper with future research directions.

2 RELATED WORK

Autonomic computing has been established as an effective method to handle the growing complexity in a wide range of application domains, from distributed storage systems (Devarakonda et al. 2003), communication networks (Dobson et al. 2006), to power control systems (Femal and Freeh 2005). In addition, a few architectural approaches were proposed to realize the vision of autonomic computing. In (White et al. 2004), for example, the authors described the interfaces and behavioral requirements for autonomic agents, the interactions between them, and several common design patterns that can be used to produce desired system-level properties. In (Tesauro et al. 2004), the authors proposed a two-layer architecture called *Unity*, where autonomic agents compute resource-level utility functions in their local application environments based on service-level attributes and resource usage patterns. The resulting utility values are used by global arbiters to optimize resource allocation between application environments on a continuous basis. Built upon the Unity architecture, a prototype was developed to demonstrate the ability of system management under competing objectives, making tradeoffs through the interaction of performance agents, power agents, and coordination agents as intermediate brokers (Das et al. 2008). Other researchers applied model-driven architectures to the development of autonomic systems, transforming system properties expressed in abstract UML models into deployable operational policies at runtime (Pena et al. 2006). Similarly, goal model decomposition was considered as a way of exploring the space of all alternative behaviors by dividing a global objective into a hierarchy of sub-objectives, which are managed by corresponding autonomic agents organized in an identical hierarchy (Lapouchnian et al. 2005).

With the rise of cloud computing paradigm, autonomic computing is gaining momentum in this area. Specifically, an autonomic mobile robot was integrated with commercial energy optimization software to help diagnose emerging thermal problems in data centers (Lenchner et al. 2011). In (Mola and Bauer 2011), the authors used hierarchical collaboration between autonomous managers to improve application response time under given service-level agreements. In (Yazir et al. 2010), the authors proposed an architecture to distribute the task of resource allocation to a group of autonomous node agents, each of which is coupled with a physical server in a data center. The node agents make VM placement and migration decisions in response to changes in their local conditions, without aiming for optimal global configuration. As shown in their experiments, this approach can attain better scalability while reducing the number of VM migrations, when compared to other methods that use global arbiters in the decision process. However, the authors did not describe how global objectives can be fulfilled in the proposed architecture. Recently, a flexible architecture called *Monalytics* was proposed in (Wang et al. 2011), which integrates continuous system monitoring with on-demand data analytics to enable cost-effective diagnosis and timely actions in support of data center management. Moreover, autonomic methods have also been used in cloud environments to address security issues such as abnormal behavior identification (Smith, Guan, and Fu 2010) and intrusion detection/prevention (Roschke, Cheng, and Meinel 2009). Although these works offer valuable insight on the design of autonomic architectures and their applicability to specific aspects of system management, none of them directly tackles the problem of bidirectional behavioral modeling in the context of virtual infrastructure management. Furthermore, most of the existing approaches adopt a hierarchical structure for online decision support, which may not be the most suitable choice for large data centers with hundreds of thousands of heterogeneous physical and virtual resources.

Symbiotic simulation is a paradigm that combines a simulation system and a physical system synergistically in a closed feedback loop (Aydt et al. 2008; Aydt et al. 2009). Driven by real-time data collected continuously from the physical system, accurate what-if scenario simulations can be conducted, which in turn help make active decisions in control of the physical system (including the measurement

process itself). This paradigm has been used in autonomic systems to achieve, for example, real-time decision making under uncertainty (Mitchell and Yilmaz 2008) and proactive traffic routing in distributed road infrastructures (Claes and Holvoet 2010). Based on the symbiotic simulation paradigm, this paper proposes an architecture for autonomic cloud management, using worker agents to make decentralized management decisions in a peer-to-peer manner while simultaneously using a complementary management network to ensure the convergence of emergent behavior towards global objectives.

3 VIRTUAL INFRASTRUCTURE MANAGEMENT STACKS

There exist many virtual infrastructure management stacks being used by different IaaS cloud providers, including proprietary ones like Amazon AWS (Amazon 2012a) and IBM SmartCloud Provisioning (IBM 2012c) as well as open source alternatives such as Eucalyptus (Eucalyptus 2012), OpenStack (OpenStack 2012), and CloudStack (Citrix 2012). This section briefly introduces the logical architectures adopted in Eucalyptus and OpenStack, providing readers with the necessary background and a basis for comparison with the multi-agent architecture proposed in the next section.

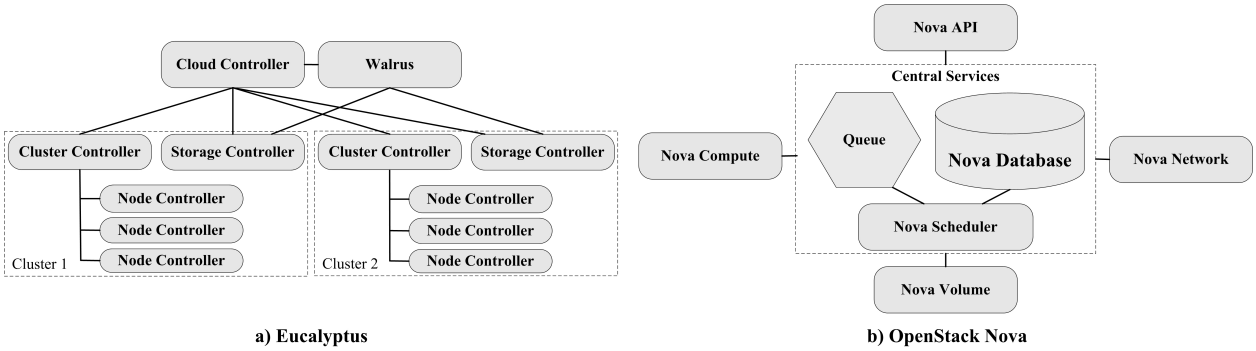

Figure 1. Logical architectures of Eucalyptus and OpenStack Nova (Eucalyptus 2012; OpenStack 2012)

Eucalyptus implements the Amazon AWS interface using a hierarchy of controllers organized at three distinct levels, as illustrated in Figure 1a. At the top level, a single cloud controller serves as the entry point for incoming requests and makes global VM scheduling decisions across multiple clusters. Additionally, it uses a single scalable storage service called Walrus to manage persistent VM images and customer data. At the middle level, a controller is created on each cluster to schedule the execution of VMs across the nodes and manage the private/public virtual networks. Each cluster also has a local storage controller to provide a block-based storage service to the cloud. At the bottom level, node controllers are attached to the physical servers within a cluster to carry out VM activities (e.g., create, execute, suspend, or terminate a VM) through the functionality provided by the hypervisors running on the nodes.

As shown in Figure 1b, the OpenStack cloud management, called Nova, is composed of a cluster of components, including API daemons for accepting customer requests and initiating the orchestration activities; compute daemons for creating and terminating VMs; volume daemons for creating, attaching, and detaching persistent storages to VMs; and network daemons for managing network connections. These daemons are coordinated by a set of central services, which use a queue for inter-daemon messaging, a database for state sharing, and a scheduler that determines the placement of VMs among the physical servers based on the current state of the entire infrastructure for efficient resource usage.

4 AN AGENT-BASED MANAGEMENT ARCHITECTURE

As shown in Figure 2, the proposed agent-based architecture consists of two interacting yet separate layers: a low-level *execution layer* and a high-level *management layer*, which correspond to the *physical* and *simulation* parts of a symbiotic system respectively. The distinction is purely logical based on the different functionality provided by each layer. In practice, the management-layer components can share the same virtual infrastructure with the other execution-layer applications and services. Alternatively, they

can be hosted in a zone dedicated to management purposes in order to provide better isolation between execution and management in a data center.

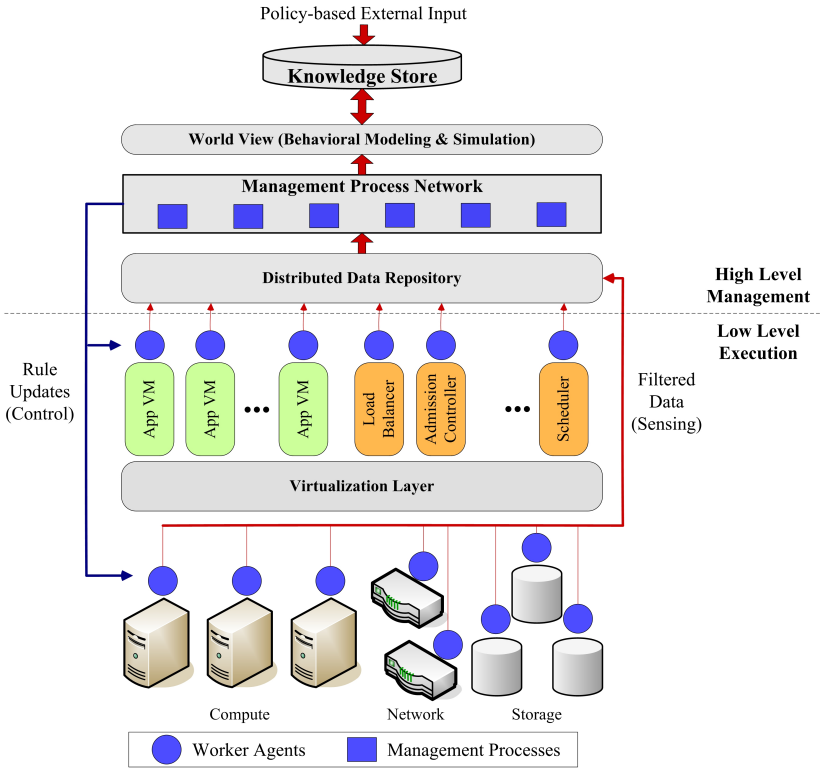

Figure 2: Logical view of agent-based autonomic management

At the execution layer, physical and virtual resources are associated with worker agents, as follows.

- **Physical resource worker agents**. For each type of physical resources (i.e., compute servers, network switches, and storage servers), a type of worker agents is defined. A worker agent is attached to an instance of the corresponding type of physical resources at bootstrap time, as a daemon process in the host OS. The runtime behavior of a worker agent is governed by a set of *execution rules* specifying such policies as which and when system metrics of interest are monitored, how to filter the collected metric data before passing them to the management processes, how to establish neighborhood relations with the other worker agents, what communication and negotiation protocols to use for inter-agent interactions, and what actions to take when a particular event occurs. To consolidate workloads, for instance, the agents can make joint decisions about live VM migration using system metrics (such as server utilization, storage availability, and network traffic condition) collected along planned migration paths, which are formed by spreading query messages to neighboring agents, assembling a list of potential source and destination servers, and mutually selecting the most suitable ones using criteria encoded in the execution rules.
- **Virtual resource worker agents**. Similarly, worker agents are also created and attached to the VMs provisioned on top of the virtualization layer, in the form of resident daemon processes in the guest OS. The VMs (and their worker agents) are divided into two groups: *application VMs* that host customer business logic (shown as App VMs in Figure 2) and *service VMs* that implement the various services of the system management stack (e.g., load balancer, admission controller, scheduler, and so forth). For the former group, the agents perform operations in the background with minimal interference with the application execution. For example, they can help scale up or scale down VMs' virtual memory dynamically as a response to changes in the observed memory usage pattern even without knowing the application logic. Conversely, the agents

in the latter group are service-aware in the sense that they are tightly integrated with the services to take advantage of the knowledge about the specific functionalities in determining their actions.

- **Execution rule definition**. To allow for agile and deterministic actions, the behavior of a worker agent is defined by a complete set of execution rules, each consisting of four components: a *pre-condition* that specifies the condition under which a particular rule will be fired, a *delay* that gives the interval between the current and the next rule evaluations in real time, a *post-condition* that defines the action to take for the current rule, and an *annotation* that captures the characteristics of the action (e.g., deadline, timeout, number of tries, etc.). These execution rules are implemented as code templates in advance using parameters that can be varied within predefined constraints, allowing the management processes to instantiate rule variations for evaluation at runtime. For instance, a load balancing rule could be programmed to use one out of several different algorithms, each with such adjustable parameters as operation scope (cluster vs. infrastructure), frequency (from once per day up to once per hour with discrete step sizes), and triggering conditions (thresholds that indicate the degree of imbalance). By combining these algorithms and parameters, a wide range of variations can be generated by management processes for scenario simulations. Note that the rules of a worker agent can also be updated by management processes dynamically in order to tune the agent's behavior based on the result of simulation analysis.

The various components at the management layer are described as follows.

- **Distributed data repository**. The worker agents publish data into a distributed repository, including *distilled metric data* that collectively capture the current state of the virtual infrastructure and *action log data* that record the actions taken by the worker agents and the resultant system state changes. These data are further analyzed by the management processes asynchronously to construct a world view for context-aware reasoning. Due to the semi-structured nature of the data and high fault tolerance requirement, a distributed storage system such as HBase (Apache 2012) or BigTable (Chang et al. 2008) would be appropriate to implement the repository, while the data analysis could be based on the MapReduce programming model (Dean and Ghemawat 2008).
- **Management network**. The key component at the management layer is a network of management processes running on a group of interconnected virtual or physical machines. As will be discussed in Section 5, the primary task of the management processes is to ensure the fulfillment of prescribed global objectives in a timely and stable manner. To do so, the management processes use machine learning techniques to extract long-term trends from the repository data, update the global world-view models (discussed next), and conduct what-if simulations to determine when and how the execution rules shall be updated at the worker agents. Using a variable number of management processes allows for exploiting parallelism during data analysis, distributed simulation of different scenarios, and/or concurrent simulation of different rule variations within each scenario. Depending on the scenario under study, the simulation can be applied to the entire infrastructure or just part of it. To provide fault tolerance and accommodate changing computational needs, management processes are allowed to join or leave the network as necessary, using standard frameworks to facilitate simulation interoperability (Chen et al. 2008).
- **World view**. The management processes perform context-aware reasoning and simulation based on a world view, which defines a set of models to represent the execution layer details with proper abstractions. At the minimum, the world view would include a *topological model* that mimics the connectivity patterns established between the worker agents, a *behavioral model* that links the conditions, actions, and effects in the decision process of individual worker agents, a *utility model* that directs the heuristic scenario analysis by assigning appropriate rewards and costs to different actions in line with prescribed global operational policies and priorities, and a *constraint model* that defines the feasible ranges in the multi-dimensional decision space. Initially, these models are provided by administrators as parameterized templates, which are subsequently updated over time either by external inputs (e.g., human administrators or expert systems) or by metrics derived from the repository data that are measured in a symbiotic fashion. The up-to-date

world-view models are also stored in the repository to be shared by all management processes.

- **Knowledge store**. When the results of the scenario analysis meet certain criteria (e.g., beyond given confidence thresholds), they are transformed and stored as reusable knowledge in a machine-processable and human-comprehensible format. The knowledge is enriched by additional data from external services and systems, such as the nature and severity of potential security threats as provided by the IBM X-Force service (IBM 2012b). To allow for effective information exchange and integration, this body of knowledge is expressed in formal conceptual ontologies (Haase et al. 2010). At runtime, the knowledge is processed by an inference engine to derive additional inferable facts, which are used by management processes to accelerate decision making.

It is worthwhile to point out several distinctive features of the proposed architecture. First, the virtual infrastructure is governed entirely by worker agents that make decentralized operational decisions in a peer-to-peer way. This not only eliminates any potential bottleneck or weak point in the architecture, but also allows the worker agents to join, leave, or fail independently without jeopardizing the infrastructure management as a whole. Secondly, each worker agent is attached to one specific physical or virtual resource. The increasing heterogeneity of the infrastructure can be addressed by defining additional types of worker agents, without complicating the logic of existing ones. Thirdly, the interaction between management processes and worker agents is indirect and connectionless, using a distributed repository to mediate upward data/state refreshment and an interest management mechanism to achieve infrequent downward rule update. Decoupling the management processes and worker agents allows for better scalability at each layer as there is no need for explicit mapping or synchronization. Finally, as will be discussed in the next section, the capability of performing bidirectional modeling and scenario-based simulation analysis at the management processes is a critical component in the symbiotic feedback loop, ensuring that the emergent behavior from individual worker agents closely follows the prescribed global management objectives.

5 BIDIRECTIONAL MODELING AND SIMULATION

At the core of the agent-based architecture is a feedback control loop between the *worker agents* and the *management process network*, allowing the latter to carry out symbiotic simulations of the former using metric data collected from the virtual infrastructure in real time. The role of the symbiotic loop is twofold. First, it provides the management processes with an up-to-date world view, enabling more accurate and reliable simulations. Conversely, the simulation results are used to guide the behavior of individual worker agents by updating their execution rules when necessary, ensuring that the worker agents exhibit a weak overall emergent behavior that converges to prescribed global objectives. Note that the management processes can also update the data collection policies encoded in the worker agents' execution rules, thus actively steering the measurement process to improve the effectiveness of data analysis at reduced computational overhead. Section 7 gives an example of such active steering of data measurement.

Emergence was discussed by Grand in his classic book (Grand 2001), while Lecky-Thompson reviewed approaches to achieving different types of emergence (Lecky-Thompson 2008). In order to successfully manage a virtualized data center using autonomic agents, weak emergence must be maintained as unexpected behavior spiraled out of control could lead to the violation of hard constraints in service-level agreements and result in a system that cannot be trusted. Hence, when runaway behavior of the worker agents is observed, it is important to be able to discern its cause and rectify it accordingly. This can be achieved in the proposed architecture by virtue of the symbiotic feedback control loop, as follows.

The management processes conduct distributed what-if scenario simulations of the world-view models to explore the space of possible behaviors of the worker agents in a systematic way. Different simulation cloning techniques can be used to achieve efficient scenario management, dynamically spawning new simulation instances at critical decision points to exploit alternative options in parallel (see, e.g., Hybinette and Fujimoto 2001; Chen et al. 2005). In the proposed architecture, these decision points could be defined as a set of *scenario events*, such as the failure/overutilization/underutilization of a physical server, a congestion at a network switch, or the detection of an attack of specific characteristic. For each scenario event, the management processes would evaluate different variations of the worker agents' execution rules

within the limits set by the constraint model, trying to find the most suitable rule definition for each type of worker agents involved. To ensure weak emergence, the simulation results of different rule variations are mapped to vectors of scalar values using functions defined in the world-view utility model. These vectors are then compared to prescribed global objectives, which are also expressed as a vector of scalar metrics. When the most desirable behavior is determined for a given scenario (i.e., the simulation produces a utility vector that is the closest match to the objective vector), the corresponding execution rules are chosen as the candidate solution for that specific event, ready to be downloaded to the worker agents.

6 CASE STUDY IN BUSINESS RESILIENCE ASSURANCE

Building a virtual infrastructure that is resilient against failures and attacks is an ongoing effort. In a book released by Google researchers (Barroso and Holzle 2009), the common sources of failures in a data center are classified into software errors (~65%), human administrative mistakes (~20%), and hardware failures (less than 10%). Due to the complex interplay between management services, a seemingly small software bug or operational mistake could result in a wave of cascading failures, rendering a significant portion of the infrastructure inoperable with severe service disruption consequences. For instance, the latest service disruption occurred at Amazon EC2 cloud was because of an incorrect change in network configuration as part of their normal service scaling activities (Amazon 2012b), whereas the disruption at Microsoft Azure cloud was caused by a leap day bug in their service code that failed to generate transfer certificates for application data encryption (Windows 2012).

Using the proposed architecture, the robustness of a virtual infrastructure can be enhanced by preventing most software- and human-related faults while mitigating the effects of hardware failures as follows.

To reduce *administrative mistakes*, the planned operations are first defined as scenarios whenever possible. These scenarios will be simulated by the management network in advance. The administrative operations are applied to the infrastructure only if the scenario simulations produce satisfactory results. Presently, many cloud providers already conduct preventative tests in separate cloud testbeds in order to verify their maintenance operations. However, such testing is often a costly and difficult task as it is almost impossible to reproduce the exact condition of the real infrastructure in a miniature testbed environment. It is our contention that this task can be facilitated by taking advantage of the modeling and data-driven simulation capability of the management network, which possesses an up-to-date world view of the operations currently happening in the cloud.

Software errors can also be reduced to a certain extent through continuous what-if simulations performed by the management network. In the proposed architecture, the management service logic is implemented by the worker agents' execution rules. The exact same rules are defined in the models of the worker agents during the simulations, albeit they are executed against models of the real infrastructure in the world view. Hence, the simulations themselves serve as automatic intensive regression tests of the management software, achieving broader coverage of the code than a human-initiated testing could attain. The coverage would only increase over time as more and more scenarios are simulated. Any suspicious bugs that might show up in the simulations are brought to the attention of software developers, allowing them to inspect and fix potential faults before they actually occur in production. During rule updates, the simulated (and properly tested) rules are directly downloaded to the worker agents without the need for code transformation, further reducing the possibility of inadvertent errors.

In contrast, *hardware failures* are handled mainly through the collaboration between neighboring worker agents, which may exchange “heart beat” messages periodically to determine the healthiness of each other. When an anomaly is detected by a worker agent, early warning messages will be propagated to all relevant agents through the neighborhood relations, enabling those agents to verify and confirm the observed error and take appropriate actions accordingly (e.g., migrating the affected VMs to other servers based on self-evacuation rules defined in the server agents or shifting network traffic to alternative routes as defined by traffic control rules in the switch agents). As these actions have already been simulated (or maneuvered) previously by the management network, this approach allows for fast response time while simultaneously minimizing potential adverse side-effects.

7 IMPLEMENTATION CHALLENGES

In order to realize the potential of the proposed architecture, a key challenge is to minimize the expected *time to action*, which is the delay between when an unanticipated event occurs in the infrastructure until it is recognized and incorporated in the next rule update at the relevant worker agents. This is necessary because, even though many scenario events can be anticipated and simulated in advance, there are occasions when anomalies not seen before emerge in production that require quick response. This challenge can be addressed by streamlining the main operations in the symbiotic feedback loop, as briefly discussed here.

- **Efficient system monitoring and data analysis**. It is important to identify a minimum set of system metrics to be collected by different types of worker agents, as well as the proper frequencies at which these metrics are measured. The metric data must provide enough information for diagnosis of anomalies, while reducing the overhead of data collection. As suggested in (Wang et al. 2011), an adaptive multi-phased monitoring approach could be used to combine *lightweight anomaly scanning at global scale* using low-cost measurement rules and *focused anomaly scrutiny* based on additional data collected from only a small subset of the worker agents around a detected anomaly area using more elaborated measurement rules. This approach also exemplifies the ability of symbiotic simulation to steer the process of data measurement interactively in accordance with changing needs of data analysis. Consequently, appropriate analytics algorithms should be chosen to consume the data incrementally over time and space, exploiting the inherent data parallelism to improve performance.
- **Prioritized behavior exploration**. The space of all possible behaviors of the worker agents under different scenarios can be too large to be explored exhaustively using a brutal force method, especially when the time to action becomes a major concern. One way to tackle this problem is through the prioritization of different rule variations so that the most valued options are explored first (e.g., ensuring application performance for premium customers at the cost of lower server utilizations). Although this approach may not always find the optimal solution, it increases the likelihood of figuring out an acceptable one within a given time limit, while still allowing for further optimization afterwards.
- **Agile adaptation to model changes.** The models in the world view are subject to constant evolution as monitoring data arrive continuously. This includes not only numerical state updates, such as fluctuation in server utilizations, but also structural model changes as worker agents are created or destroyed dynamically in the virtual infrastructure. As a result, the scenario simulations must be able to adapt to these changes in a timely fashion. To this end, incremental simulation cloning mechanisms could be useful to avoid repeated computation among independent scenarios before and after model changes (Chen et al. 2005); and existing approaches developed to support variable structure models would shed light on accommodating the dynamics of the modeling process (Uhrmacher 2001). Issues related to model adaptation in agent-based symbiotic simulations are discussed in (Kennedy et al. 2011).

8 CONCLUSION AND FUTURE WORK

Autonomic computing is gaining popularity as a promising technology to achieve self-management of virtualized cloud infrastructures. This paper proposed an agent-based info-symbiotic architecture towards this goal, using autonomous worker agents to make decentralized resource management decisions in a peer-to-peer manner. The architecture employs a network of management processes to ensure that the emergent behavior of individual worker agents converges to prescribed global objectives, performing concurrent data analysis and scenario-based simulation to determine the appropriate execution rules for different types of worker agents. Through the use of continuous system monitoring and asynchronous rule updates, a symbiotic feedback control loop is established between the management network and the worker agents to achieve bidirectional modeling. This paper also presented a qualitative case study in the context of data center business resilience assurance, arguing for the effectiveness of the proposed archi-

tecture in enhancing the robustness of virtual infrastructures. Some of the implementation challenges were discussed, along with possible approaches to addressing them in the proposed architecture.

As part of our future research, we plan to implement a prototype of the proposed architecture in open source cloud management stacks, starting from the most suitable services. In the process, we will address the challenges outlined in the previous section and evaluate different approaches quantitatively.

REFERENCES

Aydt, H., S. J. Turner, W. Cai, and M. Y. H. Low. 2008. "Symbiotic Simulation Systems: An Extended Definition Motivated by Symbiosis in Biology." In *Proceedings of the 22nd IEEE International Workshop on Principles of Advanced and Distributed Simulation*, 109-116. Roma, Italy.

Aydt, H., S. J. Turner, W. Cai, and M. Y. H. Low. 2009. "Research Issues in Symbiotic Simulation." In *Proceedings of the 2009 Winter Simulation Conference*, 1213-1222. Austin, TX.

Amazon Corporation. 2012a. Elastic Compute Cloud. Accessed May 24. http://aws.amazon.com/ec2/.

Amazon Corporation. 2012b. Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region. Accessed May 24. http://aws.amazon.com/message/65648/.

Apache Software Foundation. 2012. HBase. Accessed May 24. https://hbase.apache.org/.

Armbrust, M., A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. 2009. "Above the Clouds: A Berkeley View of Cloud Computing." Technical Report No. UCB/EECS-2009-28, University of California at Berkeley, CA.

Barroso, L. A. and U. Holzle. 2009. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. In Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers series.

Brazier, F. M. T., J. O. Kephart, H. Van Dyke Parunak, and M. N. Huhns. 2009. "Agents and Service-Oriented Computing for Autonomic Computing: A Research Agenda." *Internet Computing* 13(3):82-87.

Chang, F., J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. 2008. "Bigtable: A Distributed Storage System for Structured Data." *ACM Transactions on Computer Systems* 26(2), Article 4.

Chen, D., S. J. Turner, W. Cai, B. P. Gan, and M. Y. H. Low. 2005. "Algorithms for HLA-Based Distributed Simulation Cloning." *ACM Transactions on Modeling and Computer Simulation* 15(4):316-345.

Chen, D., G. K. Theodoropoulos, S. J. Turner, W. Cai, R. Minson, and Y. Zhang. 2008. "Large Scale Agent-Based Simulation on the Grid." *Future Generation Computer Systems* 24(7):658-671.

Claes, R. and T. Holvoet. 2010. "Maintaining a Distributed Symbiotic Relationship using Delegate Multi-agent Systems." In *Proceedings of the 2010 Winter Simulation Conference*, 2981-2990. Phoenix, AZ.

Citrix Systems. 2012. CloudStack Cloud Computing Software. Accessed May 24. http://cloudstack.org/.

Dean, J. and S. Ghemawat. 2008. "MapReduce: Simplified Data Processing on Large Clusters." *Communications of the ACM* 51(1):107-113.

Darema, F. 2005. "Grid Computing and Beyond: The Context of Dynamic Data Driven Applications Systems." *Proceedings of the IEEE* 93(3):692-697.

Darema, F., C. Douglas, and A. Patra. 2010. The Power of Dynamic Data Driven Applications Systems, *Multi-Agency Workshop on InfoSymbiotics/DDDAS*, Air Force Office of Scientific Research and National Science Foundation, August 2010, Washington DC.

Das, R., J. O. Kephart, C. Lefurgy, G. Tesauro, D. W. Levine, and H. Chan. 2008. "Autonomic Multi-Agent Management of Power and Performance in Data Centers." In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 107-114. Estoril, Portugal.

Devarakonda, M., D. Chess, I. Whalley, A. Segal, P. Goyal, A. Sachedina, K. Romanufa, E. Lassettre, W. Tetzlaff, and B. Arnold. 2003. "Policy-Based Autonomic Storage Allocation." In *Proceedings of the 14th IEEE International Workshop on Distributed Systems: Operations and Management*, LNCS 2867:143-154. Heidelberg, Germany.

Dobson, S., S. Denazis, A. Fernandez, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. 2006. "A Survey of Autonomic Communications." *ACM Transactions on Autonomous and Adaptive Systems* 1(2): 223-259.

Eucalyptus Systems. 2012. Eucalyptus Cloud. Accessed May 24. http://www.eucalyptus.com/eucalyptus-cloud.

Femal, M. E. and V. W. Freeh. 2005. "Boosting Data Center Performance through Non-Uniform Power Allocation." In *Proceedings of the 2nd International Conference on Autonomic Computing*, 250-261. Seattle, WA.

Grand, S. 2001. *Creation: Life and How to Make It*. Harvard University Press, Cambridge, MA.

Haase, P., T. Mathab, M. Schmidt, A. Eberhart, and U. Walther. 2010. "Semantic Technologies for Enterprise Cloud Management." In *Proceedings of the 9th International Semantic Web Conference*, LNCS 6497:98-113. Shanghai, China.

Huebscher, M. C. and J. A. McCann. 2008. "A Survey of Autonomic Computing – Degrees, Models, and Applications." *ACM Computing Surveys* 40(3), Article 7.

Hybinette, M. and R. M. Fujimoto. 2001. "Cloning Parallel Simulations." *ACM Transactions on Modeling and Computer Simulation* 11(4):378-407.

IBM Corporation. 2012a. SmartCloud. Accessed May 24. http://www.ibm.com/cloud-computing/.

IBM Corporation. 2012b. X-FORCE Hosted Threat Analysis Service. Accessed May 24. http://www-935.ibm.com/services/us/en/it-services/x-force-threat-analysis-service.html.

IBM Corporation. 2012c. SmartCloud Provisioning. Accessed May 24. http://www-01.ibm.com/software/tivoli/products/smartcloud-provisioning/.

Kephart, J. and D. Chess. 2003. "The Vision of Autonomic Computing." *Computer* 36(1):41-50.

Kennedy, C., G. Theodoropoulos, V. Sorge, E. Ferrari, P. Lee, and C. Skelcher. 2011. "Data Driven Simulation to Support Model Building in the Social Sciences." *Journal of Algorithms & Computational Technology* 5(4): 561-582.

Lapouchnian, A., S. Liaskos, J. Mylopoulos, and Y. Yu. 2005. "Towards Requirements-Driven Autonomic Systems Design." *ACM SIGSOFT Software Engineering Notes* 30(4):1-7.

Lecky-Thompson, G. W. 2008. *AI and Artificial Life in Video Games*. Charles River Media, Inc. Rockland, MA.

Lenchner, J., C. Isci, J. O. Kephart, C. Mansley, J. Connell, and S. McIntosh. 2011. "Towards Data Center Self-Diagnosis Using a Mobile Robot." In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, 81-90. Karlsruhe, Germany.

Mitchell, B. and L. Yilmaz. 2008. "Symbiotic Adaptive Multisimulation: An Autonomic Simulation Framework for Real-Time Decision Support under Uncertainty." *ACM Transactions on Modeling and Computer Simulation* 19(1): Article 2.

Mola, O. and M. A. Bauer. 2011. "Towards Cloud Management by Autonomic Manager Collaboration." *International Journal of Communications, Network and System Sciences* 4:790-802.

OpenStack. 2012. OpenStack Cloud Software. Accessed May 24. http://www.openstack.org/.

Pena, J., M. G. Hinchey, R. Sterritt, A. Ruiz-Cortes, and M. Resinas. 2006. "A Model-Driven Architecture Approach for Modeling, Specifying and Deploying Policies in Autonomous and Autonomic Systems." In *Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, 19-30. Indianapolis, IN.

Roschke, S., F. Cheng, and C. Meinel. 2009. "Intrusion Detection in the Cloud." In *Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing*, 729-734. Chengdu, China.

Smith, D., Q. Guan, and S. Fu. 2010. "An Anomaly Detection Framework for Autonomic Management of Compute Cloud Systems." In *Proceedings of the 34th Annual IEEE Computer Software and Applications Conference Workshops*, 376-381. Seoul, Korea.

Sotomayor, B., R. S. Montero, I. M. Llorente, and I. Foster. 2009. "Virtual Infrastructure Management in Private and Hybrid Clouds." *IEEE Internet Computing* 13(5):14-22.

Tesauro, G., D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White. 2004. "A Multi-Agent Systems Approach to Autonomic Computing." In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, 464-471. New York, NY.

Uhrmacher, A. M. 2001. "Dynamic Structures in Modeling and Simulation: A Reflective Approach." *ACM Transactions on Modeling and Computer Simulation* 11(2): 206-232.

Wang, C., K. Schwan, V. Talwar, G. Eisenhauer, L. Hu, and M. Wolf. 2011. "A Flexible Architecture Integrating Monitoring and Analytics for Managing Large-Scale Data Centers." In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, 141-150. Karlsruhe, Germany.

White, S. R., J. E. Hanson, I. Whalley, D. M. Chess, and J. O. Kephart. 2004. "An Architectural Approach to Autonomic Computing." In *Proceedings of the 2004 International Conference on Autonomic Computing*, 2-9. New York, NY.

Windows Azure. 2012. Summary of Windows Azure Service Disruption on Feb 29th, 2012. Accessed May 24. http://blogs.msdn.com/b/windowsazure/archive/2012/03/09/summary-of-windows-azure-service-disruption-on-feb-29th-2012.aspx.

Yazir, Y.O., C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady. 2010. "Dynamic Resource Allocation in Computing Clouds using Distributed Multiple Criteria Decision Analysis." In *Proceedings of the IEEE 3rd International Conference on Cloud Computing*, 91-98. Miami, FL.

Zhang, L.-J. and Q. Zhou. 2009. "CCOA: Cloud Computing Open Architecture." In *Proceedings of the 2009 IEEE International Conference on Web Services*, 607-616. Los Angeles, CA.

ACKNOWLEDGEMENT

This research is partially supported by the Industrial Development Agency (IDA) of the Irish Government. We thank Konstantinos Katrinis from IBM Dublin Research Laboratory, Ireland, for his feedback.

AUTHOR BIOGRAPHIES

QI LIU is a postdoctoral researcher at the IBM T. J. Watson Research Center in NY, USA. He received his doctorate in Electrical and Computer Engineering from Carleton University, Ottawa, Canada. His research interests include modeling and simulation methodologies and tools, parallel and distributed simulation, multicore computing, exascale systems, and cloud computing. His e-mail address is liuqi@us.ibm.com. More information at http://researcher.ibm.com/person/us-liuqi.

GEORGIOS K. THEODOROPOULOS is a senior research scientist at the IBM Dublin Research Laboratory, Ireland. Prior to joining IBM Research, he held a senior faculty position at the School of Computer Science, University of Birmingham, UK, where he founded and led the Parallel and Distributed Systems Lab and the Midlands e-Science Centre. His research interests include distributed simulation and virtual environments, complex systems, distributed systems, info-symbiotic systems, Grid and Cloud computing, and parallel computer architectures. His e-mail address is georgheo@ie.ibm.com.

DILMA DA SILVA is a researcher and manager at the IBM T. J. Watson Research Center in NY, USA. She is an ACM Distinguished Scientist and ACM Distinguished Speaker. She received her Ph.D. from Georgia Tech, USA. Her research interests include operating systems, distributed computing, cloud computing, and high end computing. Her e-mail address is dilmasilva@us.ibm.com. More information at www.research.ibm.com/people/d/dilma.

ELVIS S. LIU is a postdoctoral researcher in the School of Computer Science and Informatics, University College Dublin, and at the IBM Dublin Research Laboratory, Ireland. He received his Ph.D. from the University of Birmingham, UK, and his B.Sc. from the University of Hong Kong. His research interests include distributed virtual environments, parallel and distributed simulations, computer games, and exascale computing. His e-mail addresses are elvisliu@ie.ibm.com and elvisliu@gmail.com.