Autonomous and Energy-Aware Management of Large-Scale Cloud Infrastructures

Eugen Feller, Christine Morin

Autonomous and Energy-Aware Management of Large-Scale Cloud Infrastructures

Eugen Feller Advisor: Christine Morin
INRIA Centre Rennes - Bretagne Atlantique
Campus universitaire de Beaulieu, 35042 Rennes Cedex, France
{Eugen.Feller, Christine.Morin}@inria.fr

***Abstract*—With the advent of cloud computing and the need for increasing amount of computing power, cloud infrastructure providers are now facilitating the deployment of large-scale data centers. In order to efficiently manage such environments three important properties have to be fulfilled by their resource management frameworks: (1) scalability; (2) autonomy (i.e. self-organization and healing); (3) energy-awareness. However, existing open-source cloud management stacks (e.g. Eucalyptus, Nimbus, OpenNebula, OpenStack) have a high degree of centralization and limited power management support.**

In this context, this PhD thesis focuses on more scalable, autonomic, and energy-aware resource management frameworks for large-scale cloud infrastructures. Particularly, a novel virtual machine (VM) management system based on a self-organizing hierarchical architecture called Snooze is proposed. In order to conserve energy, Snooze automatically transitions idle servers into a low-power mode (e.g. suspend). To favor idle times the system integrates a nature-inspired VM consolidation algorithm based on the Ant Colony Optimization (ACO).

***Keywords*-Cloud Computing, Scalability, Self-Organization, Self-Healing, Consolidation, Ant Colony Optimization (ACO)**

I. Introduction

Cloud computing has recently evolved as a new computing paradigm which promises virtually unlimited resources. Customers rent resources based on the pay-as-you-go model and thus are charged only for what they use. However, customers growing demands for computing power are now facilitating the cloud service providers (e.g. Rackspace) to deploy increasing amounts of large-scale data centers. Such environments do not only impose scalability and autonomy challenges on their management frameworks but also raise questions regarding their energy-efficiency [1]. For instance, as of today Rackspace is hosting approximately 78,717 servers [2].

Several open-source cloud projects have been started to provide alternative solutions to public Infrastructure-as-a-Service (IaaS) cloud providers. Examples of such cloud management frameworks include Eucalyptus [3], Nimbus [4], OpenNebula [5], and OpenStack [6]. However, two main drawbacks exist which prevent these frameworks to efficiently manage current and future large-scale infrastructures: (1) high degree of centralization, and (2) limited support for advanced VM scheduling algorithms. While the former one leads to limited scalability and Single Point Of Failure (SPOF), the latter results in high energy costs due to resource underutilization.

In order to bridge this gap this PhD thesis proposes Snooze [7], a novel *scalable, autonomic, and energy-aware VM management framework* for private clouds. Contrary to existing works Snooze is based on a *self-organizing hierarchical* architecture and performs *distributed VM management*. Particularly, VM management is achieved by multiple managers, with each manager being in charge of a *subset* of nodes. In addition, *fault tolerance* is provided at all levels of the hierarchy. Finally, VM *monitoring and live migration* is *integrated into the framework* thus facilitating the development of advanced VM scheduling algorithms. Last but not least once idle, servers are automatically transitioned into the system administrator specified power-state (e.g. suspend) to save energy and are woken up when necessary upon new VM submission.

To favor idle times, VM consolidation can be used in order to concentrate VMs on as fewer nodes as possible. However, many of the existing consolidation approaches (e.g. [8]) adopt simple greedy algorithms such as variants of the First-Fit Decreasing (FFD) heuristic, which tend to waste a lot of resources [9] by presorting the VMs according to a single dimension (e.g. CPU). Moreover, they are known to be hard to distribute and thus have limited scalability. In this context this PhD thesis investigates distributed *nature-inspired* VM consolidation approaches to enhance scalability and proposes a novel VM consolidation algorithm [10] based on the Ant Colony Optimization (ACO) [11].

The remainder of this article is organized as follows. Section II introduces the Snooze framework. Section III presents the ACO-based consolidation algorithm. Related work is discussed in Section IV. Section V closes this article with conclusions and future work.

II. Snooze: A Scalable and Autonomic Virtual Machine Management Framework

A. System Architecture

Snooze is a VM management framework for large-scale clusters. Its architecture is shown in Figure 1. It is partitioned into three layers: *physical*, *hierarchical*, and *client*. At physical layer, machines are organized in a cluster, in which each node is controlled by a so-called *Local Controller (LC)*.

A hierarchical layer allows to scale the system, and is composed of fault-tolerant components: *Group Managers (GMs)* and a *Group Leader (GL)*. Each GM manages a *subset* of LCs and is in charge of the following tasks: (1) VM monitoring data reception from LCs, (2) Resource (i.e. CPU, memory and network utilization) demand estimation and VM scheduling,

(3) energy management, and (4) sending resource management commands (e.g. start VM, suspend host) to the LCs.

LCs enforce VM and host management commands coming from the GM. Moreover, they detect local overload/underload anomaly situations and report them to the assigned GM.

There exists one GL which oversees the GMs, keeps aggregated GM resource summary information, assigns LCs to GMs and dispatches VM submission requests to the GMs. Note, that despite the lightweight VM dispatching decisions, scalability of the GL can be further improved with replication and a load balancing layer.

To support failure detection and self-organization, multicast-based heartbeat protocols are implemented at all levels of the hierarchy. Moreover, for *performance and scalability* reasons all system components have *dedicated roles* (e.g. GL and GMs do not host VMs).

A client layer provides the user interface which is implemented by a predefined number of replicated *Entry Points (EPs)* and queried by the clients to *discover the current GL.*

In order to provide simple yet flexible interfaces, system components are implemented as Java RESTful web services. Currently, a command line interface (CLI) is implemented on top of those services. It supports the VM management as well as live visualizing and exporting of the hierarchy organization.

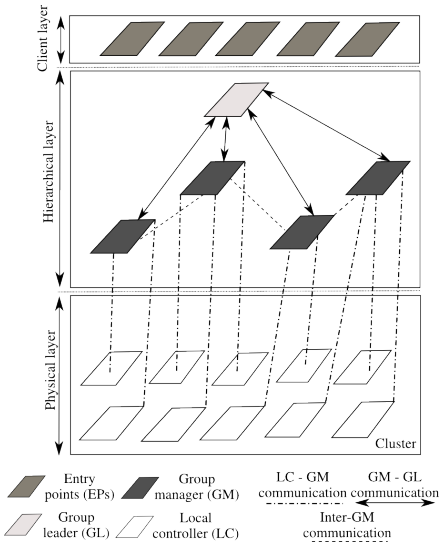

Fig. 1. System Architecture

B. Resource Monitoring and Demand Estimation

Monitoring is mandatory to take proper scheduling decisions and is performed at all layers of the system. At physical layer VMs are monitored and resource utilization information is periodically transferred to the GM by each LC. It is used by the GM in the process of VM resource demand estimation and scheduling.

At the hierarchical layer, each GM periodically sends aggregated resource monitoring information to the GL. This information includes the used and total capacity of the GM with the former being computed based on the estimated VM monitoring information of the LCs.

C. VM Scheduling

Scheduling decisions are taken at two-levels: GL and GM.

At the GL level, VM to GM dispatching decisions are taken based on the GM resource summary information. For example, VMs could be dispatched to round robin fashion or load balanced across the GMs. Note that summary information is not sufficient to take *exact dispatching decisions*. For instance, when a client submits a VM requesting 2GB of memory and a GM reports 4GB available it does not necessary mean that the VM can be finally placed on this GM as its available memory could be distributed among multiple LCs (e.g. 4 LCs with each 1GB of RAM). Consequently, a *list of candidate GMs* is provided by the dispatching policies. Based on this list, a linear search is performed by issuing VM placement requests to the GMs.

At the GM level, the actual VM scheduling decisions are taken. Therefore, four types of scheduling policies exist: placement, overload relocation, underload relocation, and finally reconfiguration. Policies of the former type (e.g. round robin or first-fit) are triggered event-based to place incoming VMs on LCs. Similarly, relocation policies are called when overload (resp. underload) events arrive from LCs and aims at moving VMs away from heavily (resp. lightly loaded) nodes. For example, in case of overload situation VMs must be relocated to a more lightly loaded node in order to mitigate performance degradation. Contrary, in case of underload, for energy saving reasons it is beneficial to move away VMs to moderately loaded LCs in order to create enough idle-time to transition the underutilized LCs into a lower power state (e.g. suspend).

Complementary to the event-based placement and relocation policies, reconfiguration policies can be specified which will be called periodically according to the system administrator specified interval to further optimize the VM placement of moderately loaded nodes. For example, a VM consolidation policy can be enabled to weekly optimize the VM placement by packing VMs on as few nodes as possible.

Note that depending on the implemented scheduling policy, scheduling decisions can be either based on resources within the scope of a single GM (i.e. centralized) or cover multiple GMs (i.e. distributed). It order to take full benefits of the system distributed policies are preferred (see Section III).

D. Self-Organization of the Hierarchy

The Snooze hierarchy self-organization works as follows. When a GM first attempts to join the system, a leader election algorithm is triggered in order to detect the current GL. Currently, our leader election scheme is built on top of the Apache ZooKeeper [12] highly available and reliable coordination system. If a leader exists, the GM joins it and starts sending GM heartbeats. Otherwise, it becomes the new GL and starts sending GL heartbeats.

On the other hand, when a LC starts it has to join the hierarchy. Therefore, information about the current GL as well as the GM to be joined is required. In order to get GL information it listens for GL heartbeats. When a heartbeat arrives, it contacts the GL to get a GM assigned. Therefore,

different LC to GM assignment policies can be enabled at the GL. For example, LCs could be assigned to GMs in round robin fashion or based on the current GM load situation (e.g. to least loaded GMs). Finally, the LC joins the assigned GM, and starts listening to GM heartbeats as well as sending own ones to its assigned GM.

E. Fault Tolerance

When a GL fails, its heartbeats are lost and the leader election procedure is restarted by one of the GMs. When an existing GM becomes the new leader it switches to GL mode and starts sending GL heartbeats. Other GMs receive the heartbeats and rejoin the new GL. LCs which were previously assigned to the failed GM fail to receive its GM heartbeats and rejoin the system.

When a GM fails, its heartbeats are lost and the managed LCs rejoin the hierarchy. Moreover, GM failures are detected by the GL based on missing heartbeats, and its contact information is gracefully removed in order to prevent new VMs from being scheduled on it.

When a LC fails, its heartbeats are lost and the GM in charge invalidates its contact information. Note, that in the event of a LC failure, VMs are also terminated. Hypervisors *snapshot* features can be used by LCs in order to periodically save VM states on stable storage on behalf of the GM. This will allow the GM to reschedule the failed VMs on its active LCs. Note, that coordinating groups of communicating VMs belonging to the same application is out of the scope of this work.

F. Evaluation

Snooze was evaluated on a 144 nodes cluster of the Grid'5000 experimentation testbed in France. Up to 500 VMs were submitted. The results [7] have shows that the fault tolerance features of the framework do not impact application performance. Moreover, negligible cost is involved in performing distributed VM management and the system remains highly scalable with increasing amounts of VMs and hosts.

III. Energy-Aware VM Management

One important goal of Snooze is to provide energy-savings in IaaS clouds. Therefore, each GM integrates mechanisms to detect idle LCs and automatically transition them in a low-power state (e.g. suspend) after a system administrator predefined idle-time threshold has been reached. Moreover, LCs are woken up by the GM in case either not enough capacity is available to handle incoming VM placement decisions or overload situations on the LCs occur.

To favor idle times, underload situations are detected by each LC and reported to the GM which then triggers the underload relocation algorithm. In addition, consolidation is performed periodically on the GMs in order to further optimize the placement of VMs on moderately loaded LCs.

A. ACO-based VM Consolidation

This PhD thesis proposes a novel nature-inspired VM consolidation algorithm [10]. The proposed algorithm is based on the ACO principles in which multiple agents (i.e. artificial ants) compute solutions probabilistically and simultaneously within multiple cycles. Thereby, they communicate indirectly by depositing a chemical substance called *pheromone* on each VM-LC pair within a *pheromone matrix*.

In each cycle the ants receive VMs, and start constructing local solutions (i.e. VM to LC assignments) by the use of a probabilistic decision rule which describes the desirability for an ant to choose a particular VM as the next one to pack in its current LC. This rule is based on the current pheromone concentration information on the VM-LC pair in the pheromone matrix and a heuristic information which guides the ants towards choosing VMs leading to better overall LC utilization. Hence, the higher the amount of pheromone and heuristic information is associated with an VM-LC pair, the higher the probability that it will be chosen.

At the end of each cycle, local solutions are compared and the one requiring the least number of LCs is saved as the new globally optimal solution. Afterwards, the pheromone matrix is updated to simulate pheromone evaporation and reinforce VM-LC pairs which belonged to the so-far best solution.

The stochastic nature of the algorithm allows it to explore a large number of potential solutions. Moreover, the algorithm is well suited for parallelization.

B. Evaluation

A centralized version of the algorithm was evaluated by simulation and compared with the well known FFD heuristic. Moreover, the CPLEX solver was used to compute the optimal solution.

Our results [10] have shown that compared to FFD, the ACO-based approach utilizes *lower amounts of hosts* and thus yields to *superior average host utilization and energy gains*. Thereby, on average 4.7% of hosts and 4.1% of energy were conserved (including energy spent into the computation). Moreover, the proposed algorithm achieves nearly optimal solutions (i.e. 1.1% deviation).

IV. Related Work

Several VM management systems such as Nimbus [4], OpenNebula [5], OpenStack [6], and Eucalyptus [3] have been developed during the last years with the former three being centralized and non fault-tolerant. Moreover, to the best of our knowledge none of the mentioned systems offers dynamic VM relocation and reconfiguration support.

Eucalyptus is the open-source system closest to ours in terms of architecture. However, it does not include any self-healing features and strictly distinguishes between cloud and cluster controllers (i.e. static hierarchy) while Snooze follows a more self-organizing approach in which each group manager (GM) is promoted to a group leader (GL) dynamically during the leader election procedure upon GL failure detection. In addition, cluster controllers in Eucalyptus are limited to simple static VM placement policies (i.e. greedy, round robin) and do not support VM live migration while Snooze supports advanced scheduling algorithms (i.e. relocation and reconfiguration) and ships with integrated live migration support.

Recently in [13] a more distributed peer-to-peer (P2P) based VM scheduling approach is introduced. However, this work is still in very early stages as it is limited to load balancing and no evaluation regarding its scalability and fault tolerance aspects is presented. Another VM management framework based on a P2P network of nodes is presented in [14]. The nodes are organized in a ring and scheduling is performed iteratively upon underload and overload events triggered by the nodes. However, neither the overhead of maintaining the ring structure nor the scalability and fault tolerance aspects are discussed. In both works only preliminary simulation-based results targeting the scheduling time are presented.

In contrast, nodes in Snooze are dynamically organized in a self-healing hierarchical architecture. This allows it to scale with an increasing number of nodes as well as to provide the required fault tolerance properties without the need to rely on P2P technology. In fact, our experimental results [7] show that our architecture is sufficient in order to provide scalability and fault tolerance properties for thousands of nodes.

Besides the existing VM management frameworks, more generic frameworks targeting scalability and fault tolerance issues in distributed systems have been proposed in the past. Particularly, the hierarchical layer of Snooze is inspired from the idea introduced in the Hasthi [15] framework which takes a hierarchical self-stabilizing approach for managing large-scale distributed systems. Contrary to Hasthi whose design is presented to be system agnostic and utilizes a distributed hash table (DHT) based P2P network, Snooze follows a simpler design and does not require the use of P2P technology. Moreover, it targets virtualized platforms and thus its design and implementation is driven by the platform specific objectives and issues. Finally, Snooze has a working implementation which was evaluated in a real environment.

V. Conclusions and Future Work

This PhD research has introduced a novel scalable, autonomic, and energy-aware VM management framework called Snooze. Unlike the existing cloud management frameworks, Snooze utilizes a *self-organizing hierarchical architecture* and *distributes the VM management tasks* across multiple group managers (GMs), with each manager having only a *subset of nodes (i.e. local controllers (LCs))*. Moreover, fault tolerance is provided at all levels of the hierarchy. Consequently, the system is able to self-heal and continue its operation despite system component failures. Finally, *VM monitoring and live migration* are integrated into the framework thus allowing Snooze to detect and react to overload and underload situations as well as facilitating the development of VM reconfiguration algorithms (e.g. dynamic consolidation). Last but not least, when energy savings are enabled, idle servers are automatically transitioned into a lower power state (e.g. suspend) and woken up on demand.

In the future, we plan to make the system even more autonomic by removing the distinction between GMs and LCs. Consequently, the decisions when a node should play the role of GM or LC in the hierarchy will be taken by the framework instead of the system administrator upon configuration.

Another important contribution of this PhD thesis is a novel nature-inspired VM consolidation algorithm based on the Ant Colony Optimization (ACO). The proposed algorithm was implemented and experimentally validated in a centralized simulation environment. The first results have demonstrated that the ACO-based approach provides superior energy gains than traditional algorithms based on the evaluated First-Fit Decreasing (FFD) heuristic and achieves nearly optimal results.

In the future we plan to integrate the proposed algorithm in Snooze. Moreover, a distributed version of the algorithm will be developed and evaluated along with the energy-saving features of Snooze under realistic workloads. Ultimately, Snooze will be open-sourced under the GPL v2 license in Spring 2012.

VI. Acknowledgment

This research is funded by the French *Agence Nationale de la Recherche (ANR)* project EcoGrappe under the contract number ANR-08-SEGI-000.

References

[1] G. International, "Make it green: Cloud computing and its contribution to climate change," 2010, http://www.greenpeace.org/usa/en/media-center/reports/make-it-green-cloud-computing/.

[2] Rackspace, "Hosting reports third quarter," 2011. [Online]. Available: http://ir.rackspace.com/phoenix.zhtml?c=221673&p=irol-newsArticle&ID=1627224&highlight=

[3] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 124–131.

[4] K. Keahey, "Science Clouds: Early Experiences in Cloud Computing for Scientific Applications," in *Cloud Computing and Its Applications 2008 (CCA-08)*, Chicago, IL, Oct. 2008.

[5] D. Milojicic, I. M. Llorente, and R. S. Montero, "OpenNebula: A cloud management tool," *IEEE Internet Computing*, vol. 15, March 2011.

[6] "OpenStack: Open source cloud computing software," 2011. [Online]. Available: http://www.openstack.org

[7] E. Feller, L. Rilling, and C. Morin, "Snooze: A Scalable and Autonomic Virtual Machine Management Framework for Private Clouds," in *Proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, May 2012.

[8] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, ser. MGC '10, 2010.

[9] T. Setzer and A. Stage, "Decision support for virtual machine reassignments in enterprise data centers," in *Network Operations and Management Symposium Workshops (NOMS Wksps)*, 2010.

[10] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proceedings of the 12th IEEE/ACM International Conference on Grid Computing*, Lyon, France, Sep. 2011.

[11] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artif. Life*, vol. 5, April 1999.

[12] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: wait-free coordination for internet-scale systems," in *USENIX Annual Technical Conference*, 2010.

[13] J. Rouzaud Cornabas, "A distributed and collaborative dynamic load balancer for virtual machine," in *Proceedings of the 5th Workshop on Virtualization in High-Performance Cloud Computing (VHPC '10) Euro-Par 2010*, Ischia, Naples Italy, 2010.

[14] F. Quesnel and A. Lèbre, "Cooperative dynamic scheduling of virtual machines in distributed systems," in *Proceedings of the 6th Workshop on Virtualization in High-Performance Cloud Computing (VHPC '11) Euro-Par 2011*, Bordeaux, France, Aug. 2011.

[15] S. Perera and D. Gannon, "Enforcing user-defined management logic in large scale systems," in *Proceedings of the 2009 Congress on Services - I*, 2009.