<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *International Conference on Cloud and Autonomic Computing (ICCAC 2015)*, Boston, Cambridge, MA, USA, 21-25 September 2015.

Citation for the original published paper:

Krzywda, J., Östberg, P-O., Elmroth, E. (2015)

A Sensor-Actuator Model for Data Center Optimization

In: *2015 International Conference on Cloud and Autonomic Computing (ICCAC)* (pp. 192-195). IEEE Computer Society

<https://doi.org/10.1109/ICCAC.2015.13>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-132425>

<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *International Conference on Cloud and Autonomic Computing (ICCAC 2015)*, Boston, Cambridge, MA, USA, 21-25 September 2015..

Citation for the original published paper:

Krzywda, J., Östberg, P-O., Elmroth, E. (2015)

A Sensor-Actuator Model for Data Center Optimization.

In: *2015 INTERNATIONAL CONFERENCE ON CLOUD AND AUTONOMIC COMPUTING (ICCAC)* (pp. 192-195). New York: IEEE Computer Society

<http://dx.doi.org/10.1109/ICCAC.2015.13>

N.B. When citing this work, cite the original published paper.

©2015 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-110297>

A Sensor-Actuator Model for Data Center Optimization

Jakub Krzywdą, Per-Olov Östberg, and Erik Elmroth

Department of Computing Science, Umeå University
SE-901 87 Umeå, Sweden

Email: {jakub, p-o, elmroth}@cs.umu.se

Abstract—Cloud data centers commonly use virtualization technologies to provision compute capacity with a level of indirection between virtual machines and physical resources. In this paper we explore the use of that level of indirection as a means for autonomic data center configuration optimization and propose a sensor-actuator model to capture optimization-relevant relationships between data center events, monitored metrics (sensors data), and management actions (actuators). The model characterizes a wide spectrum of actions to help identify the suitability of different actions in specific situations, and outlines what (and how often) data needs to be monitored to capture, classify, and respond to events that affect the performance of data center operations.

I. INTRODUCTION

Infrastructure-as-a-Service (IaaS) cloud computing environments aggregate compute resources in data centers and provision compute capacity to end-users via service-based interfaces. To facilitate a flexible and resource-agnostic view of compute resources, these environments typically abstract physical resources using some form of virtualization technology, e.g., virtual machines or process containers, and provide metered services for some form of virtual resources.

In addition to providing a flexible and resource-agnostic model for provisioning of resources, this use of virtualization technology also provides a level of indirection between physical and virtual resources that can be used to dynamically optimize the deployment and configuration of data center resources [1]. In this work we take the perspective that this optimization should be performed in an autonomic manner, with an optimization controller that continuously monitors and analyses the state of the data center, and plans and suggests optimization actions that can be executed automatically by software components (controllers) or in a semi-automated manner by system administrators [2]. In autonomic systems, this is commonly formulated as a Monitor-Analyze-Plan-Execute-Knowledge (MAPE-K) loop [3], where the optimizer encompasses the analysis and planning steps.

In order to realize such an optimizer, a model with a holistic view of the data center is needed. The model should include knowledge of not only the state of the (virtual and physical) resources but also of data center events, contextual information of resources, availability schedules, etc. In this work we address construction of a sensor-actuator model (encompassing the monitor and execute parts of the MAPE-K loop) for autonomic data center optimization, and characterize the data (sensors) and actions (actuators) that can be used in dynamic optimization of data center configurations.

The main contributions of this paper are: A classification and characterization of planned, predicted, and unpredicted events that may occur in data center operations (Section III). An analysis of the spectrum of actuators that are available to optimizers in current data centers, and a discussion of the applicability and suitability of different actions in handling of specific events (Section IV). An analysis of what (and how often) data needs to be monitored by sensors to facilitate detection and prediction of data center events (Sections V).

II. RELATED WORK

Cloud computing hypervisors such as Kernel-based Virtual Machines (KVM) [4], offer a wide variety of management actions that allow data centers to dynamically adapt infrastructure configurations using management actions (e.g., virtual machine migration, suspending physical and virtual machines). However, most proposed solutions for optimizing data center operations focus on single management actions and fail to fully leverage the potential of selection of management actions for specific situations. Nevertheless, several systems that use a substantial subset of available management actions exist. For example, Mistral [1] is a holistic control framework that optimizes trade-offs among power consumption, application performance, and adaptation costs. The Mistral controller uses the following management actions: increase/decrease virtual machine CPU capacity, add/remove virtual machines, migrate virtual machines, and power down/up physical machines. Another example of a solution that combines several management actions is Quasar [5], a cluster management system that aims to increase resource utilization while providing consistently high application performance. The system uses vertical and horizontal scaling, as well as virtual machine migration actions, to control the configuration of a data center.

For classification, some partial hierarchies of management actions exist. For example, Vaquero et al. [6] focus on scaling at the levels of server, network, and platform and provide a hierarchy of management actions related to application scalability for IaaS and Platform-as-a-Service (PaaS) approaches.

III. DATA CENTER EVENTS

In this section, we classify data center events (e.g., requests for admission of new virtual machines, workload fluctuations, or hardware failures) by predictability and discuss their impact on data center performance. Events may introduce changes in data center environments that negatively impact the performance of data center resources and applications, e.g., increased power consumption or violated SLAs. Therefore, events need

TABLE I
EVENTS CLASSIFICATION

Class	Examples of events	Approach	Speed	OH
Planned	PM power down for maintenance	Proactive	Slow	High
Predicted	Daily/weekly workload patterns	Proactive	Mid	Mid
Unpredicted	Flash crowds, hardware failures	Reactive	Fast	Low

Speed – acceptable time needed to apply changes, OH – acceptable performance overhead from applying changes.

to be understood to be able to assess what adjustments should be made to data center configurations in response. While some events are hard or even impossible to predict (e.g., hardware failures), others are more likely to happen at specific time periods (e.g., daily workload patterns). As indicated in Table I, we here classify event as planned, predicted, and unpredicted.

Planned events, e.g., powering down physical machines for maintenance, are by nature the easiest events to handle. As the time of occurrence for planned events is typically known in advance, optimizers can prepare for them before they occur. In most cases, the time available for determining the best combination of management actions (and applying them) is long, which allows the use of actions with longer execution time.

Predicted events, e.g., workload changes following a weekly pattern, can similarly to planned events be managed proactively. Actions planned to respond to predicted events may have reasonably long execution times and performance overheads, and are typically limited by how far in advance an event can be predicted. The prediction of the time when changes in resource requirements will happen and the time needed for responding to the event are both important, as they combined allow management actions to be performed just in time when needed. Such approaches minimize the time when the amount of assigned resources differs from the amount that is needed, which allows to utilize resources more effectively.

Unpredicted events, e.g., flash crowds, hardware failures, or arrivals of new services, are by definition unpredictable and need to be handled reactively. As such, management actions for unpredicted events have to be fast to apply and cannot impose high performance overhead as the physical machines in most cases are already overloaded. The key to success in reacting to unpredicted events lies in early detection of extraordinary conditions (e.g., workload spikes [7] or hardware failures) that endanger fulfilling the expected Quality of Service (QoS) or estimating the scale of changes in resource demands (for spikes) or resource availability (for failures).

IV. TAXONOMY OF ACTUATORS

Here we describe management actions that can be taken in reaction to the events discussed in Section III. We structure all considered actions into a hierarchy of actuators.

A. Physical Machine Configuration

The configuration of a physical machine can be changed by: powering a physical machine up or down, putting a physical machine into a power-saving mode (sleep mode), or dynamically changing the frequency of CPU cores.

1) *Physical Machine State*: Physical machines can be powered down to reduce the total power consumption of data centers when the computational capacity of the physical machines is not needed. Another reason for powering down physical machines is to allow maintenance work, e.g., changing a broken hard disk or upgrading the operating system. If a physical machine is not empty, i.e. hosts virtual machines, it is necessary to migrate these virtual machines to other host(s) before powering down the physical machine. Moreover, powering a physical machine up takes time and consumes energy during boot time even though no workload is processed. Therefore, using this management action requires reliable predictions about the future resource requirements to avoid costly oscillations.

2) *Power Saving Modes*: Putting a physical machine into a power-saving mode is done for similar reasons as powering down – to reduce the power consumption during decreased workload. The difference is that the boot time is shorter for power-saving modes [8], which can be beneficial when there is uncertainty about the demand for compute capacity in the near future. The lower inertia comes with a price of a higher power consumption during the power-saving mode in comparison with powered down state.

3) *Dynamic Frequency Voltage Scaling*: Scaling the frequency of a physical CPU is used to change the performance capabilities and power consumption of a physical machine [9]. Scaling down the frequency, which reduces both the power consumption and performance of a physical machine, can be used during times of decreased workload to lower the operational costs of data center. Since the time necessary to scale the frequency is very short (between nanoseconds and tens of microseconds, depending on the technique used [10]), it can be used in a reactive manner to adjust to unpredicted workload changes. However, to have that possibility, it is necessary to run at least parts of CPUs at a reduced frequency in a normal circumstances and use the maximum frequency to handle workload bursts.

B. Virtual Machine Configuration

Virtual machine configuration actions allow to change states of virtual machines and scale them.

1) *Virtual Machine State*: Changing the state of a virtual machine allows to release some computational resources that were used by that virtual machine. In the case of horizontal scaling, when scaling in an application for short time (if the next workload increase is predicted to happen in the near future), instead of killing virtual machines, they can be put in a non-active state.

Non-active states differ in the resources they release and the time that is needed to make a transition between them. We here use the following terminology for non-active states: in *Paused* state a virtual machine releases only CPU resources and stores its state in a memory, while in *Suspended* state both CPU and memory are released and the state is stored to disk.

2) *Vertical Scaling*: Vertical scaling actions allow to dynamically adjust the amount of resources assigned to already hosted virtual machines. For example, memory ballooning [11] allows to preempt memory that is not used by a virtual machine

and allocate it to virtual machines that need more memory, or return it to a pool of free resources available to newly admitted virtual machines. Also the number of virtual CPUs assigned to a virtual machine can be changed in order to control the parallelism of applications (e.g., increasing the number of virtual CPUs can allow to run more threads simultaneously).

3) *Horizontal Scaling*: Horizontal scaling actions change the number of instances of an application or a selected tier, and are particularly useful for applications that can benefit from distributing their workload among several workers. Horizontal scaling actions trigger actions of other types, e.g., scaling out causes a Virtual Machine Initial Placement action.

C. Virtual Machine-to-Physical Machine Mapping

The third category contains actions that set the mapping between virtual machines and physical machines.

1) *Virtual Machine Initial Placement*: The initial placement of a virtual machine consists of two phases: determining the placement (optimization), and enacting placement. Here we focus on the second phase and distinguish two cases of instantiation of virtual machines: booting and cloning.

In the first case, a virtual machine loads an operating system and application from an image saved in storage. Depending on the application, this can take several minutes before the application achieves full performance.

Live cloning of already hosted virtual machines is a mean to shorten the time necessary to achieve full performance of applications. There are two main approaches to live cloning: post-copy [12] and pre-copy [13] (based on the same concepts as for virtual machine migration). Because of the shorter time needed to start processing in the new location, the post-copy cloning is more suitable for handling unpredicted events. On the other hand, pre-copy seems to be more useful for events that were planned or predicted, because once the new instance is placed it does not consume any resources of the source physical machine.

2) *Virtual Machine Migration*: Virtual machine migration is an action of moving already hosted virtual machines from one physical machine (source) to another (target). It can cause an increase in power consumption and a degradation of QoS (e.g., response time) [1]. Two main types of virtual machine migration exist: live migration, which is suitable for services that should be available even during the time of migration, and cold migration for applications that can be interrupted.

The time needed for applying a virtual machine migration depends mostly on the virtual machine's memory size, network bandwidth, and the migration technique used. For live migration, similarly to VM cloning described in Section IV-C1, we distinguish post-copy and pre-copy approaches. Pre-copy is more applicable for proactive migrations, because it increases resource utilization on the source physical machine during the whole migration process. For reactive migrations, happening during unpredicted workload increases, post-copy appears to be a better choice as it moves processing away from an over-utilized physical machine immediately and later requires only sending the missing memory pages when needed.

TABLE II
MONITORED DATA CLASSIFICATION

Class	Examples	Monitoring
Static properties	Number of CPUs in PM	Triggered manually
Infrequently changing	VM-to-PM Mapping	Triggered on events
Frequently changing	CPU utilization	Constant

3) *vCPU-to-core Relationship*: vCPU-to-core Relationship actions allow to change the number of physical cores assigned to a virtual machine's virtual CPUs, as well as, to bind a virtual CPU to a particular physical core. CPU pinning may increase the performance of some applications by exploiting the temporal locality of references (data in the CPU cache) [14]. However, CPU pinning may also cause issues, e.g., limits the ability of schedulers to balance load across processors (to avoid overheating) and interferes with fair sharing policies.

V. TAXONOMY OF SENSORS

Data center parameters are monitored to facilitate detection of changes that require immediate reaction, and creation of prediction models of workload and resource consumption.

A. Relative Occurrence of Changes

Due to differences in the relative occurrence of changes, parameters require tailored ways of monitoring. Table II summarizes the differences between classes.

Static properties (e.g., the number of CPUs in a physical machine) change only in case of physical interventions of an administrator, or in the case of hardware failures. These properties do not have to be monitored constantly, but should be evaluated when changes are triggered by an administrator, or when an external system reports a failure. Static properties are used to update the data center configuration to ensure that management decisions are based on data that reflect introduced changes and detected failures.

Infrequently changing metrics describe data that change as a result of applying management actions, e.g., a virtual machine placement or a change of a physical machine state. Because of that, their update may be triggered when the management actions are started or finished.

Frequently changing metrics describe data that vary continuously, e.g., the utilization of resources at physical and virtual level, and the performance of applications. Because of the frequency of changes, metrics from this class have to be continuously monitored. Frequently changing metrics can be used for creating models describing the relationship between workload and resource utilization, for predicting future workloads, and for detecting spikes in workloads.

B. Monitored Data

We group data that needs to be monitored in three categories: physical layer, virtual layer, and application layer.

1) *Physical Layer*: The current state of physical machines is monitored to track the amount of computational resources that are currently available or can become available in longer time (e.g., in boot time for powered down physical machines). Duration of transitions between states are measured to build a

knowledge base useful for improving policies regarding powering physical machines down or putting them in a power saving mode. The frequency of CPUs is monitored and compared with the application behavior to find if there is still a possibility to improve the performance or decrease the power consumption.

Allocation of physical resources to virtual machines is monitored to keep track of available resources and acceptable overbooking levels. Utilization and performance parameters are constantly monitored to facilitate determination of optimal resource utilization levels and detection of performance bottlenecks. Power consumption shall be monitored to enable analysis of physical machine state changes and DVFS efficiency, and to capture the influence of resource utilization on power consumption.

2) *Virtual Layer*: The duration of virtual machine states is measured to build a knowledge base for predicting changes and thus improving policies of migrating and suspending virtual machines. Since the duration may depend on the virtual machine's resource utilization, workload, and application performance, measurements of these parameters shall be correlated.

Specifications of virtual machines need to be observed as they can change due to vertical scaling actions. Utilization and performance parameters at the virtual layer are constantly monitored to model resource consumption and facilitate vertical and horizontal scaling actions. Placement of virtual machines is to be tracked to explore the dependencies between the performance of virtual machines, and physical resource allocation and utilization, as well as, the effect of co-locating virtual machines on a same physical machine. Moreover, the placement of virtual machines is recorded to be able to restore them in case of a physical machine crash.

3) *Application Layer*: Continuous monitoring of workload is necessary to create a model of its changes over time. The number of arriving requests can be combined with the resource utilization at virtual machine level to form models of application behavior. These models can later be used for predictions of future workloads and resource consumption.

Monitoring throughput, response time, and application-specific key performance indicators allows to identify if a proper QoS is provided. Comparing that information with virtual machine resource specification and utilization enables to optimize (minimize) the physical resource allocation while keeping satisfactory QoS and can be used to identify proper overbooking levels.

VI. CONCLUSION

In this paper we address data center automation, and propose a sensor-actuator model for autonomic optimization of data center resource configuration based on the MAPE-K loop for autonomic systems. To construct a knowledge base for optimizers, the model characterizes different data center events, relates events to an identified set of actions that can be used to optimize data center operations (actuators), and discusses what data are needed for this type of optimization and at what timescales that data are available (sensors). The paper proposes taxonomies for selected aspects of data center configuration, elements of data center actuators, and classes of monitoring sensors. To support the analysis, the paper also

provides characterizations and discussions of the relationships between data center events, sensors, and actuators.

ACKNOWLEDGMENT

The authors would like to acknowledge Henning Groenda, Christian Stier, and Ahmed Ali-Eldin for constructive criticism and feedback on this work. This work is funded by the Swedish Research Council (VR) project Cloud Control, the Swedish Government's strategic research project eSSENCE, and the European Union's Seventh Framework Programme under grant agreement 610711 (CACTOS).

REFERENCES

- [1] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures," in *2010 IEEE 30th Int. Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2010, pp. 62–73.
- [2] P.-O. Östberg, H. Groenda, S. Wesner, J. Byrne, D. Nikolopoulos, C. Sheridan, J. Krzywdka, A. Ali-Eldin, J. Tordsson, E. Elmroth, C. Stier, K. Krogmann, J. Domaschka, C. Hauser, P. Byrne, S. Svorobej, B. McCollum, Z. Papazachos, L. Johannessen, S. Rüth, and D. Paurevic, "The CACTOS Vision of Context-Aware Cloud Topology Optimization and Simulation," in *Proceedings of the Sixth IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2014.
- [3] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.
- [4] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux virtual machine monitor," in *Proceedings of the Linux Symposium*, vol. 1, 2007, pp. 225–230.
- [5] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and QoS-aware Cluster Management," in *19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, 2014, pp. 127–144.
- [6] L. M. Vaquero, L. Roderio-Merino, and R. Buyya, "Dynamically Scaling Applications in the Cloud," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 45–52, 2011.
- [7] A. Mehta, J. Dürango, J. Tordsson, and E. Elmroth, "Online Spike Detection in Cloud Workloads," in *IEEE International Conference on Cloud Engineering (IC2E 2015)*. IEEE, 2015, pp. 446–451.
- [8] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," *Performance Evaluation*, vol. 67, no. 11, pp. 1155–1171, 2010.
- [9] H. Qian and D. Medhi, "Server operational cost optimization for cloud computing service providers over a time horizon," in *11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*. USENIX, 2011, pp. 4–4.
- [10] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *IEEE 14th International Symposium on High Performance Computer Architecture (HPCA 2008)*, 2008, pp. 123–134.
- [11] C. A. Waldspurger, "Memory Resource Management in VMware ESX Server," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 181–194, 2002.
- [12] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, "SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing," in *4th ACM European Conf. on Computer Systems (EuroSys '09)*, 2009, pp. 1–12.
- [13] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes, "Agile: elastic distributed resource scaling for Infrastructure-as-a-Service," in *USENIX Int. Conf. on Automated Computing (ICAC'13)*, 2013, pp. 69–82.
- [14] T. Klug, M. Ott, J. Weidendorfer, and C. Trinitis, "autopin – Automated Optimization of Thread-to-Core Pinning on Multicore Systems," in *Transactions on High-Performance Embedded Architectures and Compilers III*, ser. Lecture Notes in Computer Science, P. Stenström, Ed. Springer Berlin Heidelberg, 2011, vol. 6590, pp. 219–235.