Efficient Resource Management for Data Centers: The ACTiCLOUD Approach

Extended Abstract

Vasileios Karakostas
National Technical University of Athens, ICCS

Georgios Goumas
National Technical University of Athens, ICCS

Ewnetu Bayuh Lakew
Umeå University

Erik Elmroth
Umeå University

Stefanos Gerangelos
National Technical University of Athens, ICCS

Simon Kolberg
Umeå University

Konstantinos Nikas
National Technical University of Athens, ICCS

Stratos Psomadakis
National Technical University of Athens, ICCS

Dimitrios Siakavaras
National Technical University of Athens, ICCS

Petter Svärd
Umeå University

Nectarios Koziris
National Technical University of Athens, ICCS

ABSTRACT

Despite their proliferation as a dominant computing paradigm, cloud computing systems lack effective mechanisms to manage their vast resources efficiently. Resources are stranded and fragmented, limiting cloud applicability only to classes of applications that pose moderate resource demands. In addition, the need for reduced cost through consolidation introduces performance interference, as multiple VMs are co-located on the same nodes. To avoid such issues, current providers follow a rather conservative approach regarding resource management that leads to significant underutilization. ACTiCLOUD is a three-year Horizon 2020 project that aims at creating a novel cloud architecture that breaks existing scale-up and share-nothing barriers and enables the holistic management of physical resources, at both local and distributed cloud site levels. This extended abstract provides a brief overview of the resource management part of ACTiCLOUD, focusing on the design principles and the components.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**;

KEYWORDS

Resource management, resource efficiency, cloud computing, data centers, in-memory databases, NUMA, heterogeneous, scale-up/out.

SAMOS XVIII, July 15–19, 2018, Pythagorion, Samos Island, Greece

ACM ISBN 978-1-4503-6494-2/18/07…$15.00
https://doi.org/10.1145/3229631.3236095

ACM Reference format:
Vasileios Karakostas, Georgios Goumas, Ewnetu Bayuh Lakew, Erik Elmroth, Stefanos Gerangelos, Simon Kolberg, Konstantinos Nikas, Stratos Psomadakis, Dimitrios Siakavaras, Petter Svärd, and Nectarios Koziris. 2018. Efficient Resource Management for Data Centers: The ACTiCLOUD Approach. In *Proceedings of 2018 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, Pythagorion, Samos Island, Greece, July 15–19, 2018 (SAMOS XVIII),* 3 pages. https://doi.org/10.1145/3229631.3236095

1 INTRODUCTION

ACTiCLOUD [5] aims at increasing the viability of cloud deployment scenarios through enhancement of the various technology ingredients across the entire stack, i.e., the hypervisor, the cloud manager, system libraries, language runtimes, and database systems, with a novel and holistic set of mechanisms and policies. All these are built on top of new-generation computing system architectures, having a special focus on large-scale applications working on huge data sets.

1.1 Motivation

ACTiCLOUD responds to four typical scenarios of resource inefficiency in state-of-the-art cloud offerings:

Resource underutilization. Cloud service providers (CSPs) are conservative and reserve system resources for the infrequent cases of peak traffic. This strategy clearly leaves large amounts of resources underutilized [1]. ACTiCLOUD aims at improving resource efficiency and utilization through effective consolidation.

Resource unavailability. Current server architectures are unable to serve requests that exceed the resources provided by single servers. This prohibits resource-hungry applications from enjoying the benefits of cloud. ACTiCLOUD aims at applying efficient resource management on platforms that break server barriers, focusing on applications that rely on large in-memory databases.

Resource fragmentation. Despite resources being available, they might be scattered around. Hence, cloud sites are unable to host a new–potentially resource demanding–service. ACTiCLOUD aims at identifying resource fragmentation and applying efficient migration and co-scheduling policies.

Resource contention. Problems arise due to interference between applications that compete for shared resources, when these are misplaced within the cloud platform. ACTiCLOUD aims at identifying and mitigating resource interference through appropriate migration and co-location actions.

1.2 Goal & Overview

ACTiCLOUD's vision is to develop a novel cloud architecture that targets improved utilization and scalability of resources. This will ultimately translate to: (i) significant cost and performance improvements for CSPs, (ii) higher performance, stability, and lower pricing for cloud applications, and (iii) enhanced flexibility and scalability for database applications that face tough challenges trying to satisfy their resource demands in existing cloud offerings.

ACTiCLOUD adopts state-of-the-art European hardware platforms that follow two different flavors: x86-based systems that represent the typical ISA-wise component of a cloud datacenter (Numascale platform), and ARM-based systems that capture the features of low-power, microserver technologies that can serve both as cloud datacenter servers and edge devices (KMAX platform). These platforms provide aggregation support to pool resources at the rack-scale. Then, ACTiCLOUD extends an innovative light-weight hypervisor (OnApp) for virtualizing resources at the rack-scale, and improves system libraries and managed runtime systems (JVM) to support enhanced execution and resource allocation. At the orchestration layer, ACTiCLOUD: (i) embraces state-of-the-art cloud management technologies (OpenStack [9]) to utilize already existing mechanisms in resource monitoring and management, and (ii) introduces the *ACTiManager*, that undertakes the critical tasks of optimizing resource allocation, management, and utilization. Finally, ACTiCLOUD enables the efficient execution of in-memory column-store (MonetDB) and graph (Neo4j) databases, to provide novel ACTiCLOUD-enabled database-as-a-service (DBaaS) platforms, in addition to supporting traditional cloud applications through Infrastructure-as-a-Service platforms.

This extended abstract provides a brief overview of ACTiManager, focusing on the design principles, the components, and the execution lifetime of a VM within the ACTiCLOUD architecture.

2 ACTiMANAGER

ACTiManager is the most critical component in the ACTiCLOUD architecture as it plays a central role in the realization of its objectives towards next generation IaaS and DBaaS platforms. It is a novel resource management engine that complements OpenStack to optimize allocation of resources by enabling prioritization of workloads, interference mitigation, and distributed cloud site collaboration.

2.1 Design Principles

ACTiManager is designed with the following principles in mind:

Principle 1. To operate in the typical closed-loop control fashion based on: (i) monitoring and information aggregation, (ii) information processing and modeling, and (iii) decision making and actuation, leading respectively to three core components: the *Information Aggregator*, the *Modeler*, and the *Decision Maker*.

Principle 2. To be scalable in large-scale cloud installations, operating at various levels, including node, cloud site, and inter-site levels. For this reason, ACTiManager follows a modular hierarchical design, split into two sub-modules: (i) *ACTiManager.internal*, whose goal is to manage resources at a fine-grain level within the node (e.g., mapping of virtual CPUs on physical CPUs, allocation of memory, etc.), and (ii) *ACTiManager.external*, whose goal is to manage resources within and across cloud sites, enforcing high-level policies for placement, load balance, and consolidation.

Principle 3. To minimize the modifications to an existing cloud management system and to operate as an "out-of-the-box" add-on component that can be plugged-in and out of an existing installation at will. Towards this direction, we put significant effort in interfacing with existing, well established components of the core cloud manager (OpenStack), and minimizing the necessary changes required for its integration with ACTiManager.

Principle 4. ACTiManager assumes that the cloud site administration may distinguish between high-priority, latency-critical VMs (i.e., *gold* instances) and low priority, batch VMs (i.e., *silver* instances) [4, 6], potentially with a different billing policy [2].

Principle 5. ACTiManager assumes that some applications may incorporate special functionality to expose their desired metric of interest and achieved Quality of Service (QoS). If this functionality is provided by the application, ACTiManager is able to utilize it to act towards maintaining the desired QoS of applications [4].

Principle 6. To support ACTiCLOUD's strategic objectives, business scenarios, and use cases, ACTiManager relies on online characterization of the VMs and incorporates in their feature list characteristics that describe a VM's potential to suffer from or create interference (i.e., a *noisy* or a *sensitive* VM regarding the use of resources, respectively) by the co-location of another VM within the same node [3, 7, 8].

Principle 7. To dynamically identify the characteristics of the VMs, ACTiManager logically splits the site infrastructure into (i) a small number of nodes operating as *laboratory* nodes, and (ii) the rest (vast majority) of the site nodes as *production* nodes [8, 10]. More specifically, ACTiManager uses the laboratory node(s) for quickly characterizing the VMs and extracting the aforementioned information (i.e., noisy or sensitive behavior) in an execution environment that is free from any source of interference. The rest of the compute nodes, i.e., the vast majority of the resources, are treated as usual production nodes.

2.2 Components

A specific *ACTiManager.internal* instance takes care of resource orchestration internally within each node, while a single *ACTiManager.external* component orchestrates resources across nodes within a cloud site and between distributed cloud sites. Note that depending on the underlying architecture and setup of the cloud site, nodes may differ substantially across setups.

Regardless of its level of operation (node or site), ACTiManager consists of the following three components: the *Information Aggregator*, the *Modeler*, and the *Decision Maker*. More specifically, the Information Aggregator is responsible for collecting information from the various monitoring facilities. The Modeler component provides models for: (i) characterizing the execution of the applications as noisy or sensitive regarding their use of resources, (ii) detecting anomalies like interference, imbalance, overload, underload etc., (iii) predicting the impact of various actions (e.g., consolidation effect on the already executing application, migration time, and failure probability), and (iv) characterizing applications in terms of their resource footprint and co-execution behavior on a multi-tenant system. Finally, the Decision Maker component decides about the optimized resource allocation and initiates the relevant actions, including placement, prioritization, consolidation, migration, interference mitigation, and resizing of the running VMs. Note that our design and implementation of ACTiManager targets generic server platforms; however, we also pay special attention to intelligently leveraging the characteristics of the server platforms that are employed within the project, i.e., the NUMA architecture of the Numascale system, and the hyper-converged low-power heterogeneous architecture of the KMAX system.

2.3 Lifecycle of a VM in ACTiCLOUD

ACTiManager heavily relies on online monitoring and analysis of a VM's "health" status. To be able to detect any anomalies and take corrective actions, the system needs to have a solid understanding of the VMs' execution characteristics, including their performance, resource demands, required QoS levels, potential to suffer from or create interference, etc. To accomplish this, every newly spawned VM with unknown characteristics starts its execution in the laboratory node. During its execution there, ACTiManager collects monitoring information and, based on the characterization model, it creates the fingerprint of the VM, e.g., whether it is noisy or sensitive regarding resource utilization.

The VM is analyzed for a characterization period. Then, ACTiManager decides the node in which the VM will continue its execution in the production environment, by checking if the resources that the VM requires are available in any node (number of cores, memory, etc.) taking into account the priorities (gold or silver) and the characteristics (fingerprint, noisy, sensitive) of the rest of the VMs/applications that run currently in the site's nodes.

ACTiManager.internal and ACTiManager.external wake up periodically and check for various events, such as change in execution load, interference detection, and under/over-utilization, among others. In case none of the above events occur, ACTiManager keeps the current configuration and waits for the next decision period. In case any of the above events occurs, ACTiManager decides how to resolve that issue based on the various models and by applying actions at node or site level.

Finally, in case the behavior of a VM changes significantly (e.g., change in fingerprint or performance), ACTiManager may migrate that VM to the laboratory node to perform another characterization phase and extract a new fingerprint.

3 CONCLUSIONS

Improving the resource efficiency and utilization of data centers is of paramount importance due to economies of scale. Towards that goal, the ACTiCLOUD project takes an holistic approach and works on the entire stack. In this extended abstract we briefly introduced the design principles of ACTiManager, that is responsible for enforcing efficient resource management in the project's architecture.

ACKNOWLEDGMENTS

This research has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 732366 (ACTiCLOUD).

REFERENCES

[1] Luiz Andr Barroso, Jimmy Clidaras, and Urs Hlzle. 2013. *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines* (2nd ed.). Morgan & Claypool Publishers.

[2] ACTiCLOUD Consortium. 2018. Deliverable 1.2: ACTiCLOUD Architecture. (2018). https://acticloud.eu

[3] Christina Delimitrou and Christos Kozyrakis. 2013. Paragon: QoS-aware Scheduling for Heterogeneous Datacenters. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '13)*. ACM, New York, NY, USA, 77–88. https://doi.org/10.1145/2451116.2451125

[4] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-efficient and QoS-aware Cluster Management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 127–144. https://doi.org/10.1145/2541940.2541941

[5] Georgios Goumas, Konstantinos Nikas, Ewnetu Bayuh Lakew, Christos Kotselidis, Andrew Attwood, Erik Elmorth, Michail Flouris, Nikos Foutris, John Goodacre, Davide Grohmann, Vasileios Karakostas, Panagiotis Koutsourakis, Martin Kersten, Mikel Luján, Einar Rustad, John Thomson, Luis Tomas, Atle Vesterkjaer, Jim Webber, Ying Zhang, and Nectarios Koziris. 2017. ACTiCLOUD: Enabling the Next Generation of Cloud Applications. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 1836–1845. https://doi.org/10.1109/ICDCS.2017.252

[6] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. 2015. Heracles: Improving Resource Efficiency at Scale. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture (ISCA '15)*. ACM, New York, NY, USA, 450–462. https://doi.org/10.1145/2749469.2749475

[7] Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. 2011. Bubble-Up: Increasing Utilization in Modern Warehouse Scale Computers via Sensible Co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44)*. ACM, New York, NY, USA, 248–259. https://doi.org/10.1145/2155620.2155650

[8] Dejan Novaković, Nedeljko Vasić, Stanko Novaković, Dejan Kostić, and Ricardo Bianchini. 2013. DeepDive: Transparently Identifying and Managing Performance Interference in Virtualized Environments. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference (USENIX ATC'13)*. USENIX Association, Berkeley, CA, USA, 219–230. http://dl.acm.org/citation.cfm?id=2535461.2535489

[9] OpenStack 2018. Open source software for creating private and public clouds. (2018). https://www.openstack.org.

[10] Nedeljko Vasić, Dejan Novaković, Svetozar Miučin, Dejan Kostić, and Ricardo Bianchini. 2012. DejaVu: Accelerating Resource Allocation in Virtualized Environments. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVII)*. ACM, New York, NY, USA, 423–436. https://doi.org/10.1145/2150976.2151021