Information Systems 59 (2016) 98–101

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Reproducible experiments on dynamic resource allocation in cloud data centers

Andreas Wolke [a,*], Martin Bichler [a], Fernando Chirigati [b,1], Victoria Steeves [b,1]

[a] Technische Universität München, Boltzmannstraße 3, 85748 Garching, Germany
[b] New York University, United States

ARTICLE INFO

Article history:
Received 28 September 2015
Accepted 30 December 2015
Available online 7 January 2016

Keywords:
Cloud computing
Dynamic resource allocation
Reproducibility

ABSTRACT

In Wolke et al. [1] we compare the efficiency of different resource allocation strategies experimentally. We focused on dynamic environments where virtual machines need to be allocated and deallocated to servers over time. In this companion paper, we describe the simulation framework and how to run simulations to replicate experiments or run new experiments within the framework.

1. Introduction

Reproducibility is the ability of an entire experiment or study to be duplicated, and it constitutes one of the main principles of the scientific method. Research on resource allocation in cloud computing largely consists of discrete event simulations. The results are difficult to replicate and hard to compare as they are typically based on different assumptions and implementations. This not only hinders progress, but it does also not allow for reliable results expected by academics, practitioners, and other interested parties. Twenty years ago Tichy et al. [2] criticized that "the low ratio of validated results appears to be a serious weakness in computer science research."

Many academic fields have developed standards to ensure reproducibility of their experiments. For example, microeconomists developed strict guidelines on how to conduct and report experiments, which led to reliable empirical results about human behavior in economic interactions [3]. A challenge in microeconomics is the control of human subjects in a lab. In the systems literature, a challenge is the number of hard- and software components involved and the rapid technical progress of these. As in any other field of science and engineering it is still important that results can be reproduced by others and that the assumptions and details of the implementations are easily available to others. This does not only increase the credibility of the research, but it is also vital for the progress of a field.

Technology nowadays makes it possible to reveal not only the data for an experiment, but to also make it easier that others can access the simulation software such that they can reproduce the results. This article is a companion paper to [1], which reports the results of experiments on resource allocation algorithms for cloud computing infrastructures. In this companion paper, we describe the simulation software that is made available via a Docker container. We recommend readers to first read through the general experimental environment outlined in Wolke et al. [1], before reading through this paper.

In what follows, we will briefly revisit the results from [1] in the next section, describe the simulation framework, and how to run simulations in the framework. Finally, we will

DOI of original article: http://dx.doi.org/10.1016/j.is.2015.03.003
* Corresponding author.
E-mail addresses: wolke@tum.de (A. Wolke), bichler@in.tum.de (M. Bichler), fchirigati@nyu.edu (F. Chirigati), vs77@nyu.edu (V. Steeves).
[1] Reviewer.

http://dx.doi.org/10.1016/j.is.2015.12.004
0306-4379/

outline software dependencies relevant for replication experiments.

2. Experiments in Wolke et al. [1]

In Wolke et al. [1] we compare the efficiency of different resource allocation strategies experimentally. We focused on dynamic environments where virtual machines need to be allocated and deallocated to servers over time. Simple bin packing heuristics were analyzed and used to place virtual machines upon arrival. These placement heuristics can lead to suboptimal server utilization, because they cannot consider virtual machines, which arrive in the future.

We ran lab experiments and simulations with different controllers and different workloads to understand which control strategies achieve high levels of energy efficiency in different workload environments. Combinations of placement controllers and periodic reallocations achieved the highest energy efficiency subject to predefined service levels. While the type of placement heuristic had little impact on the average server demand, the type of virtual machine resource demand estimate used for the placement decisions had a significant impact on the overall energy efficiency.

These results were generated using a software framework described in this article. The same software components and implementations of different control strategies were used for lab experiments and simulations, which required the development of a new software framework. We designed our software such that other researchers can extend it, implement their own resource allocation controllers, and benchmark them against existing controllers.

3. The simulation framework

A simulation is given a set of time series as well as the server and virtual machine (VM) capacity as an input. Each of the time series describes the CPU utilization of a VM. A server's utilization is described by the sum of the VM utilizations running plus the base demand of the server itself.

An *initial placement controller* is executed in the first step of a simulation. It computes a mapping of VMs to servers which is called VM allocation. Afterwards the VMs are migrated to the server accordingly. Subsequently, the simulation loop is initiated by injecting a message to the global message pump. At the end of a simulation loop, a new message is injected to trigger the next simulation loop within 3 s. Server and VM utilization levels are updated in each simulation loop.

A *reallocation controller* is triggered by the message pump in regular intervals, potentially triggering VM migrations. Usually, these take longer than the 3 s simulation loop interval and are controlled by the message pump as well. Migrations increase the CPU and memory utilization by variable amounts on the servers involved (both migration source and target server). Appropriate values are added during the server and VM load computation.

Placement controllers allow the simulation of dynamic cloud environments where VMs are allocated and removed continuously. For a simulation, this process is described by a VM arrival–departure schedule. For each VM allocation the placement controller is executed to determine a target server for the VM. At the end of a VM's lifetime it is removed from the simulation automatically.

The simulation framework mimics the CPU and memory utilization of servers and VMs in a cloud infrastructure. Neither applications running within the VMs are modeled nor the network infrastructure. In lab experiments, real VMs and hardware components can be used instead of the simulations. The implementation leveraging a physical server infrastructure depends on micro-services (e.g., a monitoring service), which are not described in this article. Wolke [4, Appendix A] provides an overview of the experimental testbed infrastructure we used and [4, Appendix C] explains how this infrastructure was controlled by the simulation framework.

All controller implementations presented in [1] are found in the Docker container folder `SRC_BALANCER=/root/work/paper.IS2015/control/Control/src/balancer`. Initial placement controllers extend from *InitialPlacement*, reallocation controllers from *strategy.StrategyBase*, and placement controllers extend *placement. PlacementBase* classes. For new controller implementations, a class/name mapping has to be added to the `SRC_BALANCER/controller.py` script.

4. Configuring a simulation

No special configuration files exist in our framework as everything is configured within a set of Python source files in the folder `SRC_CTRL=/root/work/paper.IS2015/control/Control` with the filename prefix `conf_`.

- `SRC_CTRL/src/conf_controller.py` specifies which controllers to use for initial placement, reallocation, and placement controllers. Each value is a string that is mapped to a concrete class by the script `SRC_CTRL/src/balancer/controller.py`. Each controller requires a mapping within this script. An initial placement controller is required by all simulations while reallocation and placement controllers are optional. Controllers are disabled by setting their configuration value to *None*.
- `SRC_CTRL/src/conf_domainsize.py` contains the available domain[2] sizes with CPU and memory capacity and the probability that a domain size appears in a simulation.
- `SRC_CTRL/src/conf_domains.py` describes the number and capacity of each domain within the simulation. The setup is done procedurally and can be changed accordingly. It should be noted that the configured domains correspond to the state of a physical infrastructure if experiments are conducted.
- `SRC_CTRL/src/conf_load.py` holds a list of time series used during the simulation. Time series are stored in a special service called Times, described below, where

[2] The term domain is used as a synonym for VM within the source codes.