Performance-aware Application Distribution in the Cloud

Santiago Gómez Sáez, Vasilios Andrikopoulos, Frank Leymann
Institute of Architecture of Application Systems
Universtät Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
{gomez-saez, andrikopoulos, leymann}@iaas.uni-stuttgart.de

Abstract: The emergence of Cloud computing and the improvement of resource management techniques have contributed to an increase in the number of application developers that are strong supporters of partially or completely migrating their application to a highly scalable and pay-per-use infrastructure. In this work in progress paper we begin the analysis on how to optimally distribute the application layers in the Cloud in order to adapt its topology to handle oscillating over time workloads. More specifically, through an empirical workload analysis and characterization we present our initial evaluation of an application persistence layer's performance under different deployment scenarios.

1 Introduction

With the increasing number of available Cloud services which allow to partially or completely deploy the application in the Cloud, it becomes possible to host only some of the application components off-premise (in the Cloud), e.g. its database, while the remaining of the application remains on-premise [ABLS13]. Our investigations concentrate on optimally distributing the application in the Cloud considering the workload that the underlying infrastructure must bear. Given that the application workload oscillates over time, we therefore aim to analyze how to optimally adapt the application deployment topology accordingly. In this work in progress paper we focus on the application persistence layer and discuss our initial performance analysis for application topology options using the TPC-H database benchmark workload and data as the basis. The contributions of this work can therefore be summarized as:

1. the analysis of the relationship between the application deployment topology, its workload oscillation over time, and the expected performance,
2. a workload characterization focusing on the application persistence layer by using the TPC-H benchmark, and
3. the generation and performance evaluation of artificial workloads for different deployment topologies of the application persistence layer.

The remainder of this paper is structured as follows: Section 2 summarizes relevant concepts and related works. Sections 3 and 4 present our experiments and discuss the most important

findings, respectively. Finally, Section 5 concludes with the future work.

2 Background

Application deployment and distribution tasks in a Cloud infrastructure require the fulfillment of preliminary tasks related to compliance, underlying resources description, application distribution arrangement, cost calculations, etc. In order to guarantee the expected application performance, such tasks should incorporate performance awareness. The migration of the different layers of an application to the Cloud is analyzed in [ABLS13], where multiple migration types are categorized and their corresponding adaptation needs are identified. In [LFM+11] a migration method and tool chain based on application model enrichment for optimally distributing the application across one or multiple Cloud providers is presented. In this work we target the challenge of optimizing the application layers' distribution in the Cloud based on its workload and expected performance. Moreover, this workload may oscillate over time due to external or internal parameters. For example, an online web store workload is increased at certain time periods, e.g. before the Christmas season, which may generate unnecessary monetary costs and/or performance degradation. This problem can be analyzed from two different perspectives: from the *Cloud Consumer* and the *Cloud Provider* objectives. On the one hand, the Cloud consumer aims to maximize the resource usage while minimizing the incurred monetary costs. On the other hand, the Cloud provider's goals are associated with the utilization of virtualization and multi-tenancy techniques to minimize operational costs while ensuring isolation between the loads generated by each Cloud consumer. Our goal, therefore, is to provide the necessary methodology and artifacts to analyze such workload oscillations over time and dynamically (re-)distribute the application layers towards bridging the gap produced by the existing conflict of interests between the Cloud consumer and the Cloud provider.

According to several investigations [BM11, GRCK07, JVS98, MMVP13, MMZVP13, WMG+10], two approaches for analyzing the application workload evolution can be identified: top-down and bottom-up. In the top-down approach, the application workload is characterized, and the application behavior model is derived before or during the deployment of the application. The importance of understanding the characteristics of the application workload to achieve efficiency is discussed in [JVS98], and hence a set of micro-architectural metrics are presented. In [BM11], an application workload specification language for creating synthetic workloads to evaluate Cloud applications is created. The combination of probability distribution fitting techniques and workload parameter analysis facilitates the building of the workload behavior model. However, a drawback of only following a top-down analysis approach is the inability of handling the workload evolution over time. Bottom-up approaches address this deficiency with the help of resource consumption monitoring techniques and performance metrics. Predicting infrastructure consumption needs based on the periodicity and similarity of occurrences over time workload demands is investigated in [GRCK07], where the cyclical aspect of multiple workloads in a large number of data centers allows to characterize, predict, and place workloads. Further resource prediction analysis can be driven using simulation approaches, such in [FFH12],

by simulating the monetary and performance cost of multiple Cloud deployment options (CDO). Towards satisfying Service Level Objectives (SLOs), in [WMG+10] the probabilistic relationship between resource consumption and application performance to model the response time distribution is analyzed. The analysis and generation of performance models for data-intensive workloads in public Clouds is addressed in [MMZVP13]. Such models can ease capacity management operations to predict the workload behavior and to determine the most cost-effective resource configuration [MMVP13].

Top-down and bottom-up application workload analysis approaches can be combined over time in order to proactively satisfy application demands by dynamically (re-)adapting its topology. In this work we first focus on analyzing the application workload at the persistence layer, and therefore we use the existing TPC-H benchmark[1] as the basis to generate application workloads that fit to different probability distributions, and analyze the consequent performance under different deployment topologies.

3 Experiments

3.1 Experimental Setup

We focus our experiments on emulating the behavior of an application which is built using the three layers pattern (*presentation*, *business logic*, and *data*, i.e. persistence) proposed in [F+02]. In a first step, we generate 1GB of representative application data using the TPC-H Benchmark. In a second step, we use Apache JMeter 2.9[2] as the application load driver to emulate the application business logic layer, using the automatically generated set of 23 TPC-H SQL queries as the load. We use the following infrastructures for the distribution of the application:

- an on-premise virtualized server on 4 CPUs Intel Xeon 2.53 GHz (2 virtual cores per CPU) with 8192KB cache, 4GB RAM, and running the Ubuntu 10.04 Linux OS and MySQL 5.1.72,
- an off-premise virtualized server (IaaS) hosted in the Flexiscale Infrastructure[3] consuming 8GB RAM, 4 CPUs AMD Opteron 2GHz with 512KB cache (2 virtual cores per CPU), and running the Ubuntu 10.04 Linux OS and MySQL 5.1.67,
- and an off-premise MySQL 5.1.69 db.m1.xlarge database instance (DBaaS) hosted on Amazon RDS[4].

We create three distribution scenarios, with the application data 1) in the MySQL on-premise, 2) the MySQL on the IaaS solution, and 3) on the DBaaS solution, while the load driver remains in all cases on-premise. The application persistence layer performance

[1] TPC-H Benchmark: `http://www.tpc.org/tpch/`
[2] Apache Jmeter: `http://jmeter.apache.org/`
[3] Flexiscale: `http://www.flexiscale.com/`
[4] Amazon RDS: `http://aws.amazon.com/rds/`

Table 1: Generated TPC-H Database Schema Analysis.

Table	Size (MB)	Columns	Access Count
customer	23.41	8	8
lineitem	614.06	16	18
nation	0.0023	4	9
orders	140.49	9	12
part	23.43	9	8
partsupp	113.77	5	5
region	0.0004	3	3
supplier	1.37	7	10

in each scenario is measured by normalizing the throughput (Req./s). The measured throughput is defined as the sum of the number of transaction per second of the application database engine, including the network latency introduced by the different deployment topologies of the application persistence layer. Such experiments have been driven across 10 rounds on average per day for a period of three weeks in the last quarter of 2013.

3.2 TPC-H Characterization

As previously discussed, top-down and bottom-up approaches can be combined to evaluate and analyze the application workload and behavior over time. For this purpose, using the first deployment scenario (on-premise) we analyze the database schema and initial workload of the application as summarized by Table 1 and Table 2. Table 1 shows the relationship between the database schema characteristics and the total number of accesses that the workload performs on each table.

Table 2: TPC-H Workload Analysis.

Query	Accessed Tables	Subqueries	Total Logical Evaluations	Throughput (Req./s)			Retrieved Data (B)
				On-Premise	IaaS	DBaaS	
$Q^{(1)}$	1	0	1	0.03425	0.03396	0.03817	538
$Q^{(2)}$	5	1	13	0.07927	0.14884	3.03260	15857
$Q^{(3)}$	3	0	5	0.08687	0.11733	0.31185	376
$Q^{(4)}$	2	1	5	0.53950	0.73922	0.94903	105
$Q^{(5)}$	6	0	9	0.01148	0.02014	0.33484	130
$Q^{(6)}$	1	0	4	0.20583	0.21355	0.28261	23
$Q^{(7)}$	5	1	11	0.03123	0.04782	0.20792	163
$Q^{(8)}$	7	1	11	0.97156	1.45380	0.18196	49
$Q^{(9)}$	6	1	8	0.05947	0.09123	0.05548	4764
$Q^{(10)}$	4	0	6	0.09168	0.11970	0.49834	3454
$Q^{(11)}$	3	1	6	2.59998	4.07134	0.26802	16069
$Q^{(12)}$	2	0	7	0.21147	0.22465	0.13981	71
$Q^{(13)}$	2	1	2	0.12771	5.32350	-	16
$Q^{(14)}$	2	0	3	0.03373	0.06017	0.29052	28
$Q^{(15)}$	1	0	2	201.533	22.2591	23.1152	9
$Q^{(16)}$	2	1	2	0.11346	0.11219	0.13471	120
$Q^{(17)}$	3	1	6	0.10931	0.19021	0.97148	648259
$Q^{(18)}$	2	1	5	0.98213	1.81212	-	25
$Q^{(19)}$	3	1	3	-	-	-	-
$Q^{(20)}$	2	0	25	4.05648	4.90228	0.17083	21
$Q^{(21)}$	5	2	8	3.02705	5.32847	-	8989
$Q^{(22)}$	4	2	13	0.01070	0.01734	0.06065	8944
$Q^{(23)}$	2	2	6	2.72083	3.30785	-	137
Avg.	3.17	0.73	7	9.89262	2.29976	1.72467	32188.5
Median	3	1	6	0.12058	0.20188	0.27531	133.5
σ	1.74	0.68	5.23	41.8356	2.67162	5.23164	137693.78

A secondary analysis consists of dissecting the set of queries which constitute the workload, and quantitatively analyzing their logical complexity, table joints, subqueries, etc. (Table 2). Throughput and retrieved data size measurements are considered as performance metrics, and therefore are a part of the bottom-up analysis approach. We combined both analysis approaches towards analyzing the relationship between the complexity of the workload queries and the performance of different application persistence deployment topologies. Towards this goal, queries are categorized by trimming the mid-range of the initial workload measured throughput and by comparing the total number of logical evaluations with respect to the remaining set of queries in the workload. Given the strong connection between the measured throughput and the resource consumption of the database engine in the TPC-H benchmark, the following categories are defined: compute high (CH), compute medium (CM), and compute low (CL).

3.3 Workload Generation and Evaluation

Subsequent to characterizing and grouping the application workload into the previously presented categories, workload generation and evaluation can take place. For this purpose, probabilistic workload generations techniques are supported, for example, by Malgen[5] and Rain [BLY+10]. The former supports the creation of large distributed data sets for parallel processing benchmarking, e.g. Hadoop, while the latter provides a flexible and adaptable workload generator framework for Cloud computing applications. However, such approaches do not consider the distribution of the application layers.

By using scripting techniques and the set of successfully executed TPC-H queries, we created multiple workloads for each of the categories we identified in the previous with size 1000 SQL queries each, using a different probability distribution for each workload. The workload probability distribution was derived by calculating its cumulative function and using the probability distribution fitting functionalities provided by EasyFit[6]. Table 3 summarizes the measured average throughput for each distribution. In the future, we plan to evaluate existing workload generation tools to incorporate support for generating multiple artificial workloads according to specified probability distributions considering the different deployment topologies of the application layers.

4 Analysis & Discussion

The empirical analysis depicted in Tables 1 and 2 drives the following conclusions with respect to the initial workload:

- the number of accesses on the different tables is not proportionally distributed across the queries constituting the workload,

[5]Malgen: `http://code.google.com/p/malgen/`
[6]EasyFit: `http://www.mathwave.com/`

Table 3: Workload and Application Data Distribution Evaluation Results.

Scenario	Category	Probability Distribution	Throughput (Req./s)
On-Premise	CL	Inv. Gaussian	0.27749
On-Premise	CM	Pareto	0.05888
On-Premise	CH	Lognormal	0.02696
DBaaS	CL	Log-Logaristic	0.45238
DBaaS	CM	Inv. Gaussian	0.19972
DBaaS	CH	Lognormal	0.10273

- the queries which constitute the initial workload are highly heterogeneous with respect to the logical complexity and to the retrieved data, and
- such heterogeneity is also observed when comparing the throughput fluctuation of the different queries among the analyzed scenarios.
- When deploying the database in the IaaS solution, the average performance of 85% of the successfully executed queries improves between 3% and 4078%. However, such maximum peak is due to the nature of query $Q^{(15)}$, an SQL *CREATE VIEW* statement. In average, the initial workload performance without considering the maximum and minimum peaks is improved in approximately 166%.
- When deploying the database in the DBaaS solution, the average performance of 70% of the successfully executed queries improves between 11% and 3725%. However, such maximum peak is related again to the nature of query $Q^{(15)}$, and the average performance without considering the maximum and minimum peaks improvement of the initial workload is 225%.
- There are queries whose performance is degraded when being executed off-premise, such as $Q^{(1)}$, $Q^{(15)}$, and $Q^{(16)}$ for the IaaS solution scenario, and such as $Q^{(8)}$, $Q^{(9)}$, $Q^{(11)}$, $Q^{(12)}$, $Q^{(15)}$, and $Q^{(20)}$ for the DBaaS solution scenario.

After analyzing the individual behavior of the queries that constitute the initial workload, we analyze the performance of the generated workloads under different deployment topologies of the application persistence layer. We can observe from the obtained results depicted in Table 3 that:

- the compute demand is indeed increased among the three different workload categories, and the throughput is reduced by 78% to 90% when executing the workload on-premise, and by 55% to 57% when executing the workload in a DBaaS solution, using the CL category as the baseline, and

- the overall performance is highly improved when executing the generated workloads in a DBaaS solution, observing an increase of 163%, 339%, and 381% for the CL, CM, and CH workloads, respectively.

From the previous results it can be therefore concluded that not only different workload distributions perform in a different manner, but also that adapting the application deployment topology with respect to the workload demands significantly and proactively improves the application performance. We therefore observe a necessity of providing support for (re-)adapting the application topology, i.e. (re-)distributing its layers to optimally consume the required resources to satisfy the workload demands oscillations. With the help of workload characterization and generation techniques, probabilistic models, and prediction capabilities, the application can be proactively and optimally adapted to satisfy different workload demands. Focusing on the application persistence layer, the combination of data replication techniques with dynamic routing of requests may help to achieve an optimal performance. Dynamic routing of database requests has been previously studied and enabled by the CDASMix [GALS10] approach which allows transparent access to multiple databases without requiring adaptations to the other application layers. CDASMix can therefore facilitate the (re-)distribution of the persistence layer across on- and off-premise solutions.

5 Conclusions and Future Work

Through the experimental analysis discussed in this work in progress paper, we demonstrate that the different application layers can produce oscillating over time workloads which impact on the overall application performance. The application layers (re-)distribution is proposed as an approach to proactively ameliorate performance degradations. Focusing on (re-)distributing the application persistence layer by means of evaluating on- vs. off-premise deployment models, and analyzing, characterizing, and generating workloads with different characteristics, our experiments already show an enhanced performance when the persistence layer is deployed off-premise. This improvement however varies in each distribution scenario depending on the type of queries used and the workload distribution.

Future work focuses on analyzing the performance for further deployment topologies. In order to help application developers to derive its topology and to (re-)adapt and tune it over time, we also plan to extend existing application topology description approaches to include support to proactively meet fluctuating application performance demands.

6 Acknowledgments

This work is funded by the FP7 EU-FET project 600792 ALLOW Ensembles.

References

[ABLS13] Vasilios Andrikopoulos, Tobias Binz, Frank Leymann, and Steve Strauch. How to Adapt Applications for the Cloud Environment. *Computing*, 95(6):493–535, 2013.

[BLY+10] Aaron Beitch, Brandon Liu, Timothy Yung, Rean Griffith, Armando Fox, and David A. Patterson. Rain: A Workload Generation Toolkit for Cloud Computing Applications. Technical Report UCB/EECS-2010-14, University of California, Feb 2010.

[BM11] Arshdeep Bahga and Vijay Krishna Madisetti. Synthetic Workload Generation for Cloud Computing Applications. *JSEA*, 4:396–410, 2011.

[F+02] M. Fowler et al. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.

[FFH12] Florian Fittkau, Sören Frey, and Wilhelm Hasselbring. CDOSim: Simulating cloud deployment options for software migration support. In *Proceedings of MESOCA'12*, pages 37–46. IEEE, 2012.

[GALS10] Santiago Gómez Sáez, Vasilios Andrikopoulos, Frank Leymann, and Steve Strauch. Evaluating Caching Strategies for Cloud Data Access using an Enterprise Service Bus. In *Proceedings of IC2E'14*, 2010. *(to appear)*.

[GRCK07] Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. In *Proceedings of IISWC'07*, pages 171–180, 2007.

[JVS98] Lizy Kurian John, Purnima Vasudevan, and Jyotsna Sabarinathan. Workload Characterization: Motivation, Goals and Methodology. In *Proceedings of WWC'98*, 1998.

[LFM+11] Frank Leymann, Christoph Fehling, Ralph Mietzner, Alexander Nowak, and Schahram Dustdar. Moving Applications to the Cloud: An Approach based on Application Model Enrichment. *IJCIS*, 20(3):307–356, October 2011.

[MMVP13] Rizwan Mian, Patrick Martin, and Jose Luis Vazquez-Poletti. Provisioning Data Analytic Workloads in a Cloud. *FGCS*, 29:1452–1458, 2013.

[MMZVP13] Rizwan Mian, Patrick Martin, Farhana Zulkernine, and Jose Luis Vazquez-Poletti. Towards Building Performance Models for Data-intensive Workloads in Public Clouds. In *Proceedings of ICPE'13*, 2013.

[WMG+10] Brian J. Watson, Manish Marwah, Daniel Gmach, Yuan Chen, Martin Arlitt, and Zhikui Wang. Probabilistic Performance Modeling of Virtualized Resource Allocation. In *Proceedings of ICAC'10*, 2010.

All links were last followed on October 20, 2008.