Online Replica Placement in Cloud Environments

Avrilia Floratou
IBM Almaden Research Center
aflorat@us.ibm.com

Navneet Potti
University of Wisconsin-Madison
nav@cs.wisc.edu

Jignesh M. Patel
University of Wisconsin-Madison
jignesh@cs.wisc.edu

Database-as-a-Service (DaaS) providers need to provide performance and availability guarantees to their customers, typically in the form of Service Level Objectives (SLOs). Replication is a key mechanism that is used to meet availability targets. A critical challenge is to meet a target availability goal while minimizing the total operating cost. In multi-tenant environments where each tenant needs only a fraction of the resources of a single node (e.g., in [1]), the degree of multi-tenant concurrency per node is high, which makes guaranteeing the performance SLOs challenging.

Another challenge is that the DaaS providers typically have an *estimate* of the workloads that they expect to serve, but the *actual* workload characteristics may deviate from this estimate. An online data placement algorithm (as opposed to offline techniques, such as [2])) tackles exactly this situation – it finds a placement for the replicas of a given tenant as soon as the tenant arrives into the system, without assuming a priori knowledge of the actual workloads of the entire set of tenants. Thus, the online placement techniques gracefully adjust to unexpected workload changes when the actual workload is different from the expected workloads.

We examine the online replica placement problem in the DaaS setting. We assume that the tenant's database is replicated a few times so that the availability SLOs are met. In the model that we consider, there is a master/primary replica that drives the load on the slave/secondary replicas (e.g., as in [1]), by forwarding certain operations to them.

Designing a replica placement algorithm for multi-tenant DaaS environments is a challenging task, since the algorithm must: a) consider the different performance requirements of all the tenants, b) consider the differences in the load between the replicas of the same tenant, c) not violate the replication constraints, d) aim to balance the load across all machines, e) minimize the total operating cost, and f) gracefully adjust to unexpected workload changes.

We examine a number of replica placement algorithms that can be used in such a multi-tenant setting. More specifically, we study adaptations of the traditional bin-packing algorithms (such as *First Fit*), as well as extensions to the well known *Round Robin* algorithm. We also present a new algorithm, called *RkC*, which is based on the notion of the `power of two random choices` [3].

We evaluate the algorithms using a set of four criteria that cover a wide spectrum of the needs of cloud service providers. More specifically, we examine the impact of the algorithm on the number of machines needed in order to accommodate a set of tenants with various performance requirements, the load distribution across these machines under normal conditions and in case of a node failure, as well as the adaptivity of the algorithms in case of unexpected workload fluctuations.

Our results show that the RkC algorithm has many appealing aspects, including: a) it has low cost since the number of machines it uses is very close to the theoretical lower bound, b) it is able to evenly spread the primary and the secondary replicas across the cluster, c) it is able to place the secondary replicas in such a way that the load can be balanced in case of machine failures, d) it can accommodate rack constraints, e) it is flexible enough to accommodate workload changes, f) the above properties hold for a variety of replication factors, number of tenants and machine capacities.

Acknowledgments

This research was supported in part by the National Science Foundation under grant III-0963993.

References

[1] Philip A. Bernstein et al., "Adapting microsoft sql server for cloud computing," in *ICDE*, 2011, pp. 1255–1263.

[2] Carlo Curino et al., "Workload-aware database monitoring and consolidation," in *SIGMOD Conference*, 2011, pp. 313–324.

[3] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, 2001.

Submission to SoCC '14, October, 2014, Seattle, WA, USA.

Online Replica Placement in Cloud Environments

Avrilia Floratou[1], Navneet Potti[2], Jignesh M. Patel[2]
[1]IBM Almaden Research Center, [2] University of Wisconsin-Madison

Database-as-a-Service (DaaS) providers need to provide performance and availability guarantees to their customers, typically in the form of Service Level Objectives (SLOs).

Replication is a key mechanism that is used to meet availability targets.

A critical challenge is to meet a target availability goal while **minimizing the total operating cost**. In multi-tenant environments where each tenant needs only a fraction of the resources of a single node, the degree of multi-tenant concurrency per node is high, which makes guaranteeing the performance SLOs challenging.

How to place each tenant's replicas in a Database-as-a-Service setting so that the tenant performance SLOs are satisfied?

Challenges

1) The cloud service providers are not aware of the actual workload characteristics that they expect to serve.
2) The service providers' workload estimates may significantly deviate from the actual workload.

We focus on online replica placement algorithms that do not assume a priori knowledge of the actual workload of the entire set of tenants.

a) We assume that the tenant's database is replicated a few times so that the availability SLOs are met.
b) There is a master/primary replica that drives the load on the slave/secondary replicas, by forwarding certain operations to them.
c) The tenants are split into tenant classes, and each class has its own performance SLO. For example, if there are two classes, then one class may have a 100tps (transactions per second) SLO, whereas the other class may have a 10tps SLO.

Online Replica Placement Challenges

The online replica placement algorithm must:
a) consider the different performance requirements of all the tenants,
b) consider the differences in the load between the replicas of the same tenant,
c) not violate the replication constraints,
d) aim to balance the load across all machines,
e) minimize the total operating cost, and
f) gracefully adjust to unexpected workload changes.

Online Replica Placement algorithms

We designed various online replica placement algorithms:

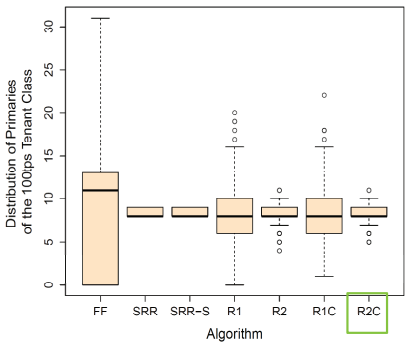
a) FF: First Fit adjusted for replication constraints and performance SLOs

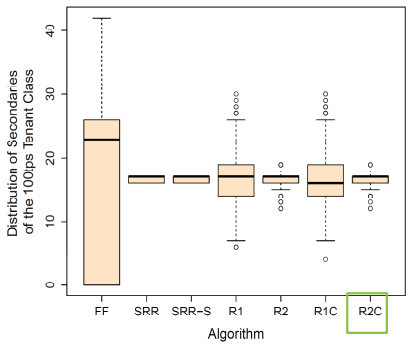
b) SRR, SRR-S: Round Robin algorithms adjusted for replication constraints and performance SLOs

c) Rk, RkC: Novel algorithms based on the power of two random choices.

**RkC is the winner in all the metrics:
Number of machines used, adaptivity in workload changes, load balancing and load balancing after a node failure**

Load Balancing





Load Balancing After a Node Failure