Predicting In-Memory Database Performance for Automating Cluster Management Tasks

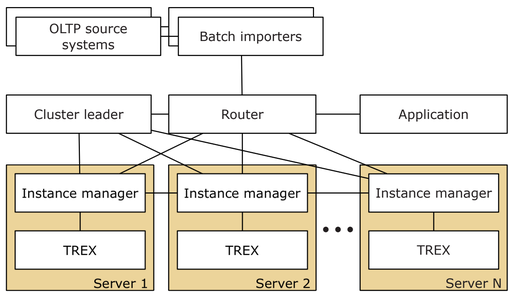
Jan Schaffner[1], Benjamin Eckart[1], Dean Jacobs[2], Christian Schwarz[1], Hasso Plattner[1], Alexander Zeier[1]

[1]Hasso Plattner Institute, University of Potsdam, Germany [2]SAP AG, Walldorf, Germany

Motivation and problem setting

Enterprise SaaS solutions commonly maintain data in a farm of conventional databases. To reduce total cost of ownership, multiple businesses are consolidated into each database instance (multi-tenancy). The database farm is exposed to continuous variations in the request rates of tenants, which are due to diurnal as well as longer-term usage patterns. Variations in request rates necessitate performing administrative tasks such as expanding or contracting the cluster and migrating tenants to balance the work across the servers.

A high degree of automation is required to avoid manual intervention by administrators. Automating cluster management requires profound knowledge about the effect that placing a tenant on a given server will have on that server's ability to handle requests according to our SLO (i.e. response times of 1000 ms in the 99-th percentile).

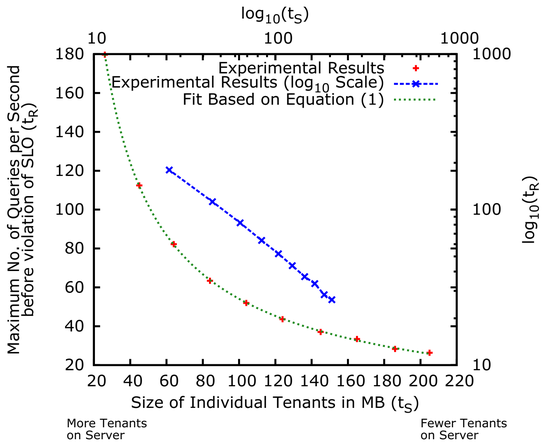


The Rock clustering framework runs in front of a collection of TREX servers (SAP's in-memory column database) and provides multi-tenancy, replication of tenant data, and fault tolerance. Read requests are submitted to the cluster by an analytics application. Write requests are submitted by batch importers, which periodically pull incremental updates of the data from transactional source systems. The Rock framework has been used for all experiments presented here.

Goal: Predicting the combined load on a server when consolidating multiple tenants onto it

We are building an emprical model, therefore a black box view of the database is adopted and observations about the response times are made based on experiments. The experimental results are then generalized into a model which captures important aspects of the experiment configuration (e.g. the request rate) as parameters.

Workload as a function of request rate and size



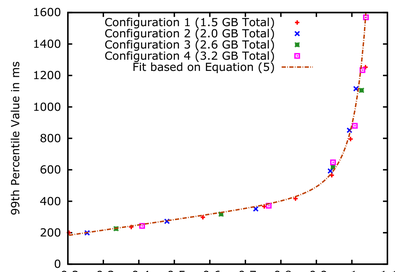
$$f(t_S) = a \cdot {t_S}^b + c \quad \textbf{(1)}$$

$$1 = \frac{a \cdot {t_S}^b + c}{t_R} \quad \textbf{(2)}$$

$$w(t_S, t_R) = \frac{a \cdot {t_S}^b + c}{t_R} \quad \textbf{(3)}$$

Using Equation (1), we can compute the request rate per tenant such that the response time goal is not violated for any tenant on the server. Our goal is to calculate the *workload* incurred by a tenant on a server given arbitrary values for size and request rate. Therefore, Equation (3) defines a function such that w=1 denotes the point where the 99th percentile value exceeds the response time goal. This can be done by dividing Equation (1) by the request rate.

Decreasing request rate or size would result in a function value of w<1 and the server would be unlikely to violate the response time goal. But can Equation (3) be used to predict 99-th percentile values less than 1000 ms? Is it additive across multiple tenants with different request rates and sizes (a.k.a configurations)?



Smaller size & same request rate ≈ same size & smaller request rate:

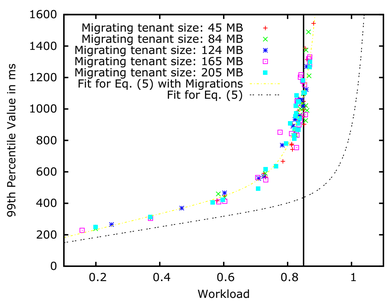
$$w(T) := \sum_{t \in T} \frac{a \quad {t_S}^b + c}{t_R} \text{ , with } T = \quad (t_S^1, t_R^1), \ldots, (t_S^n, t_R^n) \quad \textbf{(4)}$$

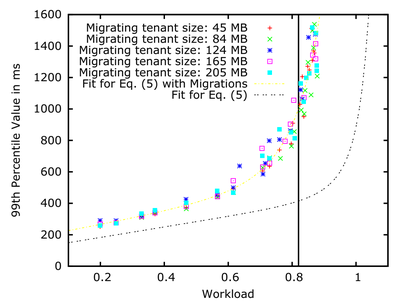
Predict response time in 99-th percentile for arbitrary workload:

$$f(w) = a \cdot w + b \cdot \exp(c \cdot w^d) + e \quad \textbf{(5)}$$

Extending the model with migrations

When reacting to changes in the load pattern of one or more servers in the cluster, it might become necessary to adapt the assignment of tenants to servers in the farm. However, the migration of tenants between servers consumes resources that would otherwise be available for query processing. Especially when migration is being performed in response to an overload situation on a server, SLO violations might occur as a direct consequence of the migration cost.

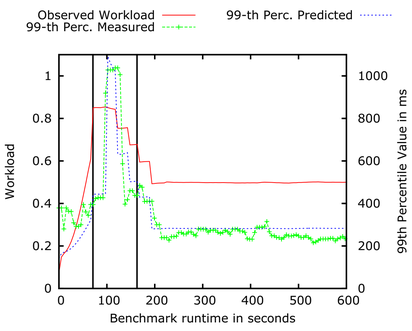


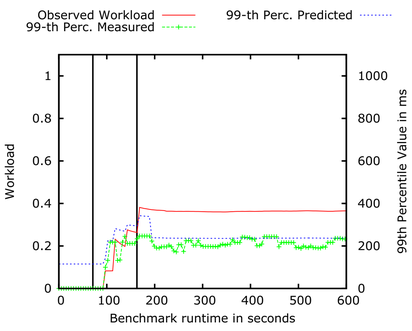


Automatic migration based on workload changes

The previous chart shows that the size of the tenant being migrated does not affect how much capacity is lost during migration, given that only one tenant per server is being migrated at a time. Tenant size does, however, affect the duration of a migration.

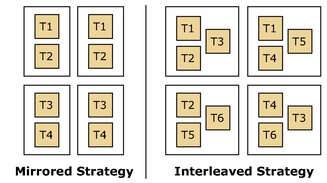
We developed a migration algorithm that uses the workload model to automatically react to server overload and underload conditions. In contrast to data placement algorithms that reorganize the entire cluster all at once, only incremental changes to the layout are made (orchestrated by the cluster leader).





Extending the model with failures

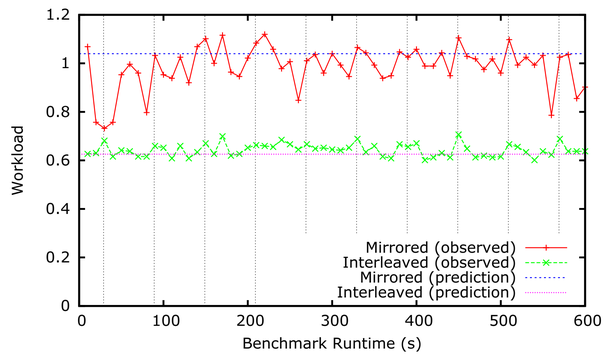
Replication of data provides the following advantages in our context: more opportunities for distributing work across the cluster; masking server outages, both planned (upgrades) and unplanned (failures); and masking resource-intensive administrative operations such as migrations and the merge of deltas into columns. We study two static placement strategies, *mirrored* and *interleaved*.



Maximum number of concurrent users without SLO violation:

	Mirrored	*Interleaved*	*Improvement*
Normal operations	4218 users	4506 users	7%
Periodical failures	2265 users	4250 users	88%

Workload pattern for worst server in the cluster:



HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam