BRB: better batch scheduling to reduce tail latencies in distributed data stores

Keywords : Data centers; Load-balancing; Batching; Tail latency.

Waleed Reda, Lalith Suresh, Marco Canini, Sean Braithwaite

Abstract* – *A common pattern in the architectures of modern interactive web-services is that of large request fan-outs, where even a single end-user request (task) arriving at an application server triggers tens to thousands of data accesses (sub-tasks) to different stateful backend servers. The overall response time of each task is bottlenecked by the completion time of the slowest sub-task, making such workloads highly sensitive to the tail of latency distribution of the backend tier. To address such challenges, we present BetteR Batch (BRB), a system that carefully schedules requests in a decentralized and task-aware manner. In doing so, we improve performance predictability of data stores in the presence of large request fan-outs. We show via simulations that our proposed design improves latencies over the state-of-the-art by a factor of 2.

Introduction

Recent studies [1] showed that latency distributions in Web-scale systems exhibit long-tail behaviors where the 99th percentile latency can be more than one order of magnitude higher than the median latency. To make matters worse, modern applications are highly distributed. For instance, interactive web services involve parallelization and aggregation of responses across 10s-1000s of servers, all of which need to finish for an end-user request (e.g., a search query) to be considered complete.

There have been several proposals for achieving latency reduction and lowering the impact of skewed performance across backend servers. These include (i) duplicating or re-issuing requests, predicting stragglers, or trading off completeness for latency [1], (ii) policy-based resource allocation and admission control (with the objective of achieving fairness or satisfying SLOs) [2], and recently (iii) making use of adaptive load-balancing [3]. In this work, we present BetteR Batch (BRB), which complements the aforementioned approaches using task-aware scheduling — a method of scheduling tasks across stateful backend replica servers according to expected service time of the enclosed sub-tasks.

Task-aware Scheduling Algorithms

A key insight to reduce tail latencies is that a task's response time depends on the last of its requests to complete. BRB is based on a class of algorithms for task-aware scheduling that exploit this fact. At a high-level, these algorithms run at the clients of the data store as follows. When receiving a task, clients subdivide it into a set of sub-tasks, one for each replica group; therefore, a sub-task contains all requests for a distinct replica group. Clients then determine the bottleneck sub-task based on the costliest sub-task and assign a priority to every request in the task. Then, the priority information is propagated to the servers, which based on this can decide what request to serve next. To this end, we have designed two simple yet effective priority assignment algorithms:

1. ***EqualMax:*** Requests are given the same priority as that of the bottleneck sub-task. The intuition is that tasks with shorter bottlenecks should be given precedence in order to minimize their makespan.
2. ***UnifIncr:*** Requests are ranked based on the difference between the cost of the bottleneck sub-task and their individual cost. In other words, this effectively delays non-urgent requests in favor of bottlenecking requests.

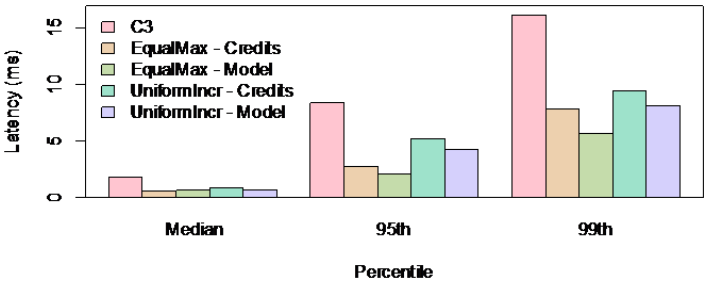

Figure 1: Task latency comparisons between BRB strategies and C3 [3].

Preliminary Evaluation

We resort to simulation to evaluate the potential benefits of our proposal. We assess BRB's latency reductions versus ideal as well as state-of-the-art approaches — in our case, C3 [3]. In an ideal case, referred to as ***model***, servers utilize a work-pulling mechanism to fetch requests from a single global priority-based queue shared by all clients. However, such a model is unrealizable since it assumes perfect knowledge of global state. Hence, we develop a ***credits*** strategy where clients report their demands at measurement intervals and are assigned credits (i.e., shares of server capacity) proportionally to demands via a logically-centralized controller.

For our evaluation, we use a real workload gathered from SoundCloud comprised of 0.5 million tasks and then generate task inter-arrival times using a Poisson process where the mean rate is set to match 70% of system capacity. Figure 1 depicts the read latencies averaged across experiments for different percentiles. The standard deviation is not shown as it is largely negligible. As shown, the credits strategy is at most 38% of an ideal model across different simulation settings. In addition, BRB outperforms C3 across all percentiles for both EqualMax and UnifIncr and improves the latencies by up to a factor of 3 at the median and 95th percentiles and up to 2 times at the 99th percentile.

References

[1] J. Dean and L. A. Barroso. "The Tail At Scale". Communications of the ACM, 56(2):74-80, 2013.

[2] D. Shue, M. J. Freedman, and A. Shaikh. "Performance Isolation and Fairness for Multi-tenant Cloud Storage". In OSDI, 2012.

[3] L. Suresh, M. Canini, S. Schmid, and A. Feldmann. "C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection". In NSDI, 2015.