Self tuning data-stores for geo-distributed cloud applications

Shankaranarayanan P N (student)
Purdue University

Ashiwan Sivakumar (student)
Purdue University

Sanjay Rao
Purdue University

Mohit Tawarmalani
Purdue University
{*spuzhava, asivakum, sanjay, mtawarma@purdue.edu*}

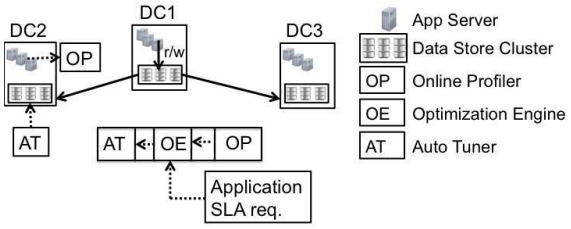

Figure 1: System Design

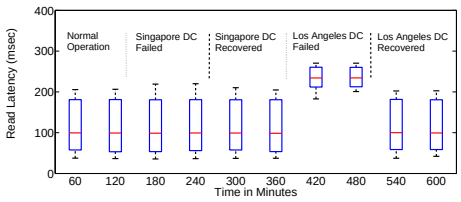

Figure 2: Box plot showing the read latency under normal condition and failure of different replicas.

Introduction: [1] Modern internet applications have resulted in users sharing data with each other in an interactive fashion. These applications have very stringent service level agreements (SLAs) which place tight constraints on the performance of the underlying geo-distributed data-stores. Deploying these systems in the cloud to meet such constraints is a challenging task, as application architects have to strike an optimal balance among different contrasting objectives such as maintaining consistency between multiple replicas, minimizing access latency and ensuring high availability. Achieving these objectives requires carefully configuring a number of low-level parameters of the data-stores, such as the number of replicas, which DCs contain which data, and the underlying consistency protocol parameters. In this work, we adopt a systematic approach where we develop analytical models that capture the performance of a data-store based on application workload and build a system that can automatically configure the data-store for optimal performance.

System design: Figure1 shows the architecture of our system. The OE takes in the workload characteristics and data-store performance measurements from the OP and applies the application SLA requirements, to generate optimal configuration strategies as the output. These configurations are then pushed to the data-store by the AT, which is a middleware library. The design makes the system components transparent to the application as well as the data-store, and scale in an elastic fashion in the cloud.

The OE comprises of many analytical models that we developed to capture the performance of the data-store. The model objectives are to optimize the latency for different percentile of requests and to generate configuration decisions at a lower granularity (group of data-items rather than the entire database). Our models also exploit the diversity in the access locations, asymmetry in access patterns and relative priority between reads and writes to enhance the performance of the system.

Evaluation: The models used in our initial experiments are built based on Cassandra's [2] architecture. We conduct experiments on a Cassandra cluster (27 nodes, each corresponding to the AWS Edge locations [1]) on an EC2 testbed using real-world traces of Wikipedia and Twitter. Figure2 shows the read latency observed from one of our experiments on a Wiki article for which reads and writes arrive from all over the world. The OE chooses replicas in US and Asia (with a quorum of size 2). We learn from the results that failure of some replicas can degrade the performance and thus it is very critical to optimize for latency under failure of DCs. This insight enable us to develop models that consider availability of DCs when optimizing for latency.

As future work, our implementation would focus on (i) the scalability (and overhead) of our system (ii) deciding the right granularity of configuration decisions (iii) dynamically adapting to the workload shifts and (iv) ensuring stability of the system by preventing oscillations. We believe that our work is one of the first that helps geo-distributed applications to automatically tune the data-store for optimal performance.

References

[1] Aws edge locations. `http://aws.amazon.com/about-aws/globalinfrastructure`.

[2] LAKSHMAN, A., AND MALIK, P. Cassandra:a decentralized structured storage system. *Newsletter. ACM SIGOPS Operating Systems Review 44* (2010), 35–40.

[1] We plan to present only a poster