Pegasus: Load-Aware Selective Replication with an In-Network Coherence Directory

Jialin Li[1], Jacob Nelson[2], Xin Jin[3], and Dan R. K. Ports[2]

[1]University of Washington, [2]Microsoft Research, [3]Johns Hopkins University

1 Overview

Modern distributed storage systems are tasked with providing fast, predictable performance in spite of immense and unpredictable load. Workload skew presents a formidable challenge for these system: some stored objects receive millions of requests per day while others see almost none. Traditional load balancing approaches (e.g., consistent hashing) fall short here, forcing datacenter operators to run the system at lower utilization.

Recent research has shown that programmable switches can provide load balancing guarantees by caching popular items in the switch dataplane [2]. However, this approach presents two problems. First, hardware resource limitations make in-dataplane storage infeasible in many scenarios: all cached data must fit in limited switch memory, and value size is limited to what can fit in the packet header vector. In production environments, existing routing functionality already strains the limits of these resources. Second, write-through caching approaches are only effective on read-mostly workloads.

In this talk, we will present a new approach to load balancing, Pegasus, that circumvents these challenges. Pegasus is based on a co-design of a distributed storage system with an in-network load balancing layer. It selectively replicates popular data items, using a programmable switch to manage object locations. A key design goal is that the switch stores only a small amount of metadata, not data contents.

2 The Pegasus Approach

Pegasus is an co-designed architecture for a rack-scale storage system, consisting of multiple storage servers and a programmable ToR switch that connects them. The system *selectively replicates* popular objects to multiple storage servers. The switch uses an *in-network coherence directory* to track object locations, and implements *load-aware forwarding*.

Selective Replication. Pegasus leverages the key insight behind NetCache and other systems: that provable load balancing can be achieved by relieving the load on only a small number of objects ($n \log n$ in the number of servers) [1]. Rather than achieve this by caching, Pegasus instead *replicates* the most popular objects to multiple servers. Conceptually, if these objects are replicated to all servers, and requests are always sent to the least-loaded server, there is always sufficient capacity to process requests. By reducing the replication factor on objects with low read-to-write ratio, this approach works for write-intensive workloads as well.

In-Network Coherence Directory. Pegasus implements selective replication using a P4 switch. Drawing inspiration from CPU cache coherency protocols, the Pegasus switch acts as a *coherence directory* that tracks which objects are replicated and where. The directory tracks only the $O(n \log n)$ most popular objects, and maps them to a list of servers.

The switch uses the coherence directory to perform content-based routing. Read requests for a popular item are forwarded to the least loaded replica for that item; those for non-popular items are forwarded using a standard consistent hashing policy. On write requests, the switch selects a new replica set from the least loaded servers, effectively rebalancing the system on every write. The routing implementation includes two additional noteworthy techniques: (1) It uses a version-based coherence protocol to ensure linearizability of requests even when messages are reordered in the network, and (2) It tracks server load using a combination of server-reported load telemetry and switch-based load prediction.

3 Results

We have implemented a prototype of Pegasus on a Barefoot Tofino switch. In brief, our evaluation with 32 servers shows:

- Pegasus reduces the 99th-percentile latency for skewed workloads by up to 97%.
- Pegasus can increase the throughput by up to 9× – or reduce by 88% the number of servers required – of a system subject to a 99%-latency SLO.
- Pegasus can react quickly to dynamic workloads where the set of hot keys changes rapidly.
- Pegasus provides the same benefits as NetCache while consuming 1/200th the switch memory (less than 5 Kb), easing practical deployment
- Pegasus is able to achieve these benefits for many classes of workloads, both read-heavy and write-heavy, with different levels of skew.

References

[1] B. Fan, H. Lim, et al. Small cache, big effect: Provable load balancing for randomly partitioned cluster services. In *SOCC '11*.

[2] X. Jin, X. Li, et al. NetCache: Balancing key-value stores with fast in-network caching. In *SOSP '17*.