Zoolander: Efficient Latency Management in NoSQL Stores

Aniket Chakrabarti¹, Christopher Stewart¹, Daiyi Yang² and Rean Griffith³
The Ohio State University¹, StumbleUpon.com² and VMWare³

Naive teacher
Mrs. Predictable, please prepare an example of your student’s work for Parent Night.

Yes, sir. I will ask Max to do it. He is my best student.

Smart teacher
Mrs. Predictable, please prepare an example of your student’s work for Parent Night.

Yes, sir. I will make every student do a project. Then I will pick the best for display.

Comedic Analogy

Surprise! Max picks the weekend of Parent Night to try something new, forgetting to do his homework.

Why isn’t my kid learning anything here? I want free tuition.

I want to try something new, Zoolander creates the best project, even though his last few projects were not very good.

Good results. Of course, Parents are not happy. They just are not angry with Mrs. Predictable.

Poor results. Of course, Parents are not. They are just angry.

How about the food at Parent Night? It was awful. I want it to be better!

NoSQL Stores – No Latency Surprises

• Narrow APIs for data access; get(key) or put(key,val)
• Scale throughput via scale out
• In practice, 10⁹ accesses per day is common
• Low latency for every access remains challenging
  • Service level objectives now include latency
    • “99% of NoSQL puts must complete within 15ms”
  • Key Problem: Periodic background jobs unexpectedly tie up resources, creating delays

Latency Surprises are not Surprising

Service times in key value stores exhibit heavy tails
  • Typical response times are very fast (1ms)
  • Type of request (get or put) has little effect
  • Occasional delays are relatively HUGE
    • 50ms to 1ms buffer dump; 500ms for DNS issue

Figure 2: We setup Apache Zookeeper[1] on 3 nodes in a private cluster. We issued 100K writes, no concurrency

The slowest 10% accounted for 20% of total delay.

It’s not just Zookeeper!
– 99.9th Percentile in Google BigTable is 31X the mean[2]

Zoolander: Middleware for NoSQL Stores

1. Target SLO is an input
  – Zoolander meets very strict, low latency SLOs
  – 99.99% of requests within 15ms
  – Orders of magnitude better than the state of the art

2. Selectively Use Replication for Predictability
  – Send all requests to all nodes, take first response
  – Adding nodes improve predictability, not throughput
  – Mitigate surprises

Zoolander Manager

SLO is 98% of puts within 15ms
  • How do we reach 99% or 99.99%? Add nodes!

Zoolander Manager adds nodes carefully, Model Driven
  • Efficient policy, i.e., meet SLO using fewest nodes
  • Should we partition?
    • Not under low utilization
    • Diminishing returns
  • Should we replicate for predictability?
    • Only if it is clearly better that partitioning

Large-Scale Case Study

Result: Zoolander reduced SLO violations by 32%, but used twice as many servers.

Map-Reduce Services:
– SLO violations force managers to keep map/reduce nodes on pay-as-you-go clouds

Scientific Computing Services:
– Moving to the cloud but use barriers heavily
  • Each violation delays the whole workload (recall, delays are heavy tailed)

References