Zoolander: Efficiently Meeting Very Strict, Low-Latency SLOs

Christopher Stewart and Aniket Chakrabarti
The Ohio State University

Rean Griffith
VMWARE

Funded in part by NSF EAGER 123077

Cost Effective Scaling

1. Traditional Scale Out
2. Big Data Challenges
3. The Time has Come
4. Contributions

- For Internet services, slow response times cost

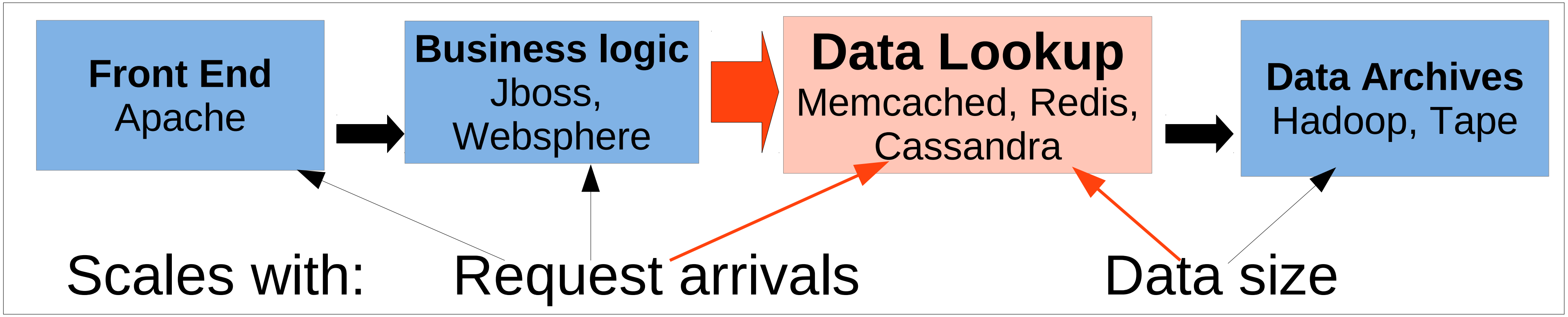
Google 100-400ms delay reduces searches per session [Google '09]

WAL★MART **100ms delay drops revenue by 1% [Crocker et al. '12]**

- Revenue **>>** Hardware Costs
  - To profit: Revenue > Hardware + Salaries + Benefits etc.

Cost Effective Scaling

- As arrival rate grows, processing tiers scale out
- As data grows, data tiers scale out



- **In big-data era, frequent data access per request**
  - TripAdvisor: each request causes 20-40 memcached accesses [Gelfond, 2011]
  - Map-reduce services and graph processing issue 10^3--10^5

Cost Effective Scaling

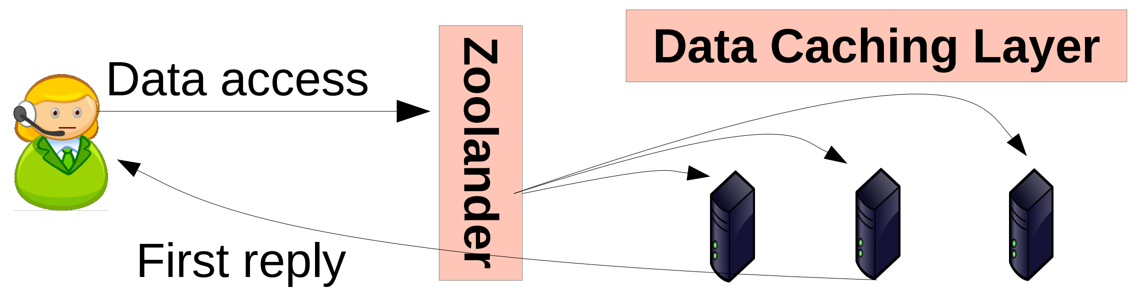
- Each user request sees 99th percentile
  - **1 slow outlier out of 100 causes 1% revenue drop**
- Service level objective: Ensure that 99.9% of data accesses complete within 15ms
- Traditional scale-out approaches struggle to reach such strict, low-latency SLOs
  - Slow response times cost 2.6B in lost sales (about 2% of market cap) [Flaherty,2012]

Cost Effective Scaling

Replication for predictability is a dumb idea whose time has come --- Line borrowed from [Mogul, 2003]

Scaling out via replication for predictability:

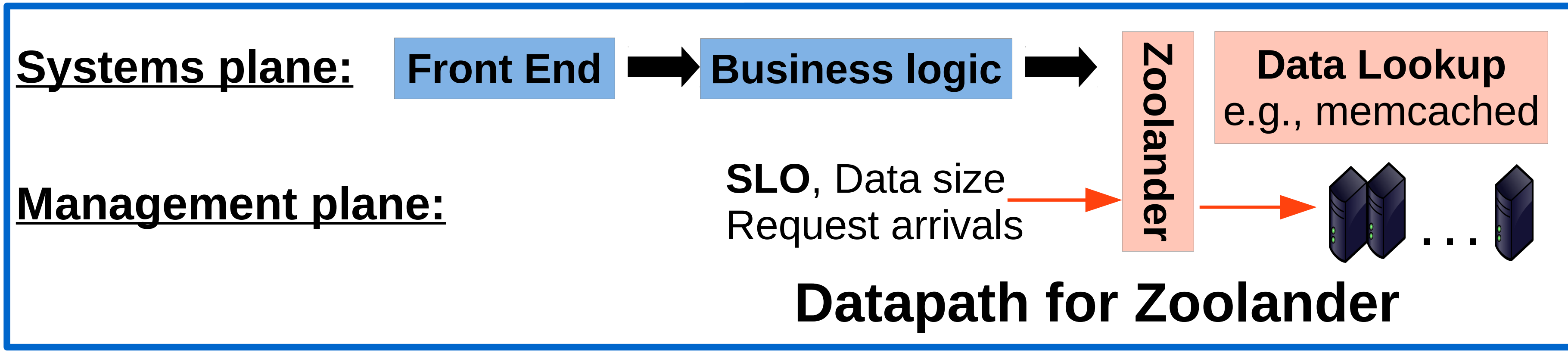
Naive Approach:
Replicate data to D nodes
Send accesses to all D
Take first response



- Old, dumb idea → more resources ≠ more throughput
- Time has come → more resources = stronger SLO

Cost Effective Scaling

1. Traditional Scale Out
2. Big Data Challenges
3. **The Time has Come**
4. Contributions



- Zoolander is middleware for key value stores
  - Meets strict SLOs efficiently using **traditional approaches and replication for predictability**

- **This talk:** Modeling and managing SLOs
  - *New way to think about predictability & scale out*

Cost Effective Scaling

- **Not this talk, but in the paper**
- **Zoolander contributes novel system designs**
  - Treat existing stores as PODS for scale out and full read/write
  - Reuses existing code & features (e.g., fault tolerance)
  - Hi-bandwidth reads reuse existing replicas for fault tolerance
  - Persistent TCP connections and fast-read bypass for low overhead
  - Support a range of consistency semantics: Causal consistency [NSDI '13], Read your own write, and eventual

Cost Effective Scaling

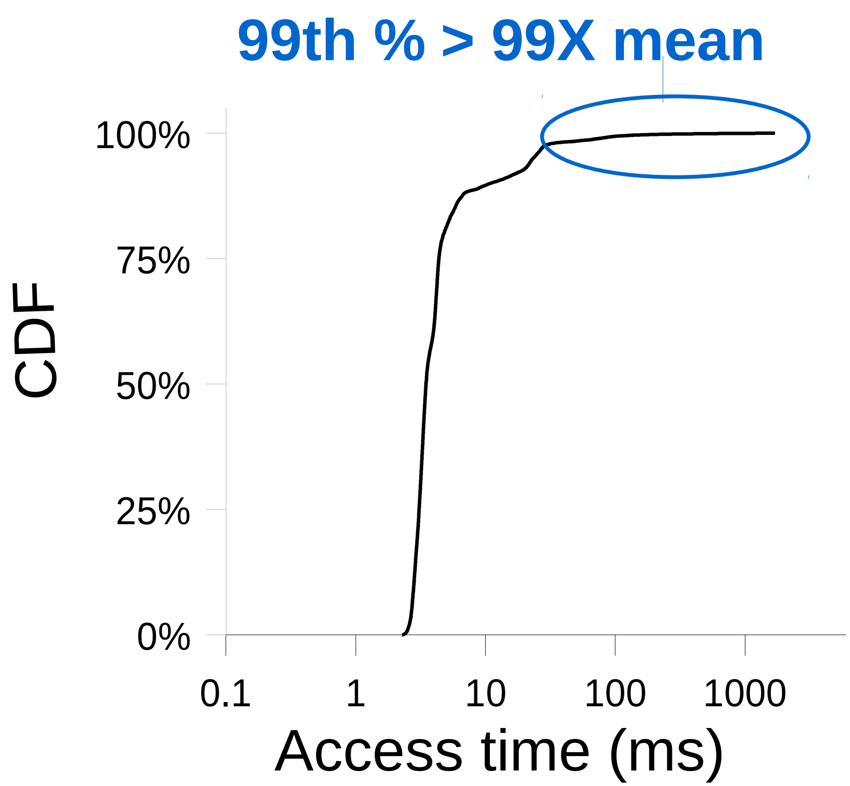
- **Revived under many aliases in recent literature:**
  - Replication for predictability [Trushkowsky, FAST '10]
  - Cloning [Ananthanarayanan, NSDI '13; Dean, OSDI '04]
  - Redundant execution [Dean & Barroso, Comm ACM '13]
- **Things they do that Zoolander doesn't:**
  - Wait for timeout and resend [Dean & Barroso, Comm ACM '13]
    - *Our model extends to this case*
- **Things Zoolander does that they don't:**
  - Scale out to D duplicates, support consistent writes, manage SLO

Cost Effective Scaling

- Can we use replication for predictability to meet strict SLOs?
  - Study access-time tails in key value stores
  - Model replication for predictability on SLOs
- Should we scale out this way?
  - Model-driven study: Rep. for pred. vs Other approaches
  - Case study: Zoolander at scale

Fat Tails in Key Values

1. Statistical Properties
2. Core Model
3. Model Validation

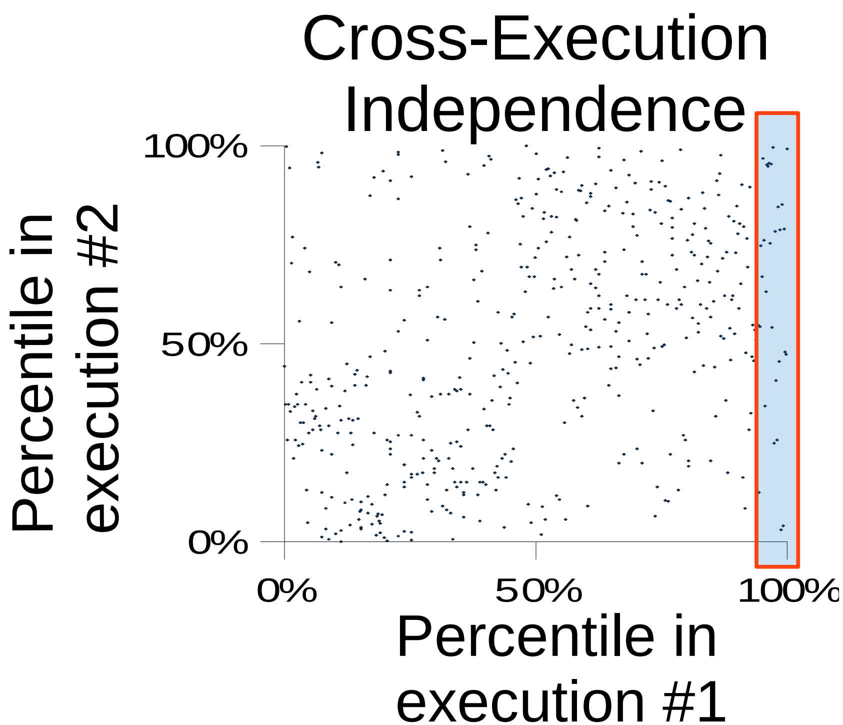


3-node Zookeeper on 4 core 2.4Ghz, data size = 1 GB, 100K writes issued serially

- Fat/Heavy Tail: Outliers are way out; not captured by normal or exponential distributions
- Org. BigTable: 99.9th percentile was 31X mean [dean '12]
- Same result: memcached, Redis, Cassandra; private, EC2
- Root cause: OS, background jobs, and performance bugs

Fat Tails in Key Values

1. Statistical Properties
2. Core Model
3. Model Validation

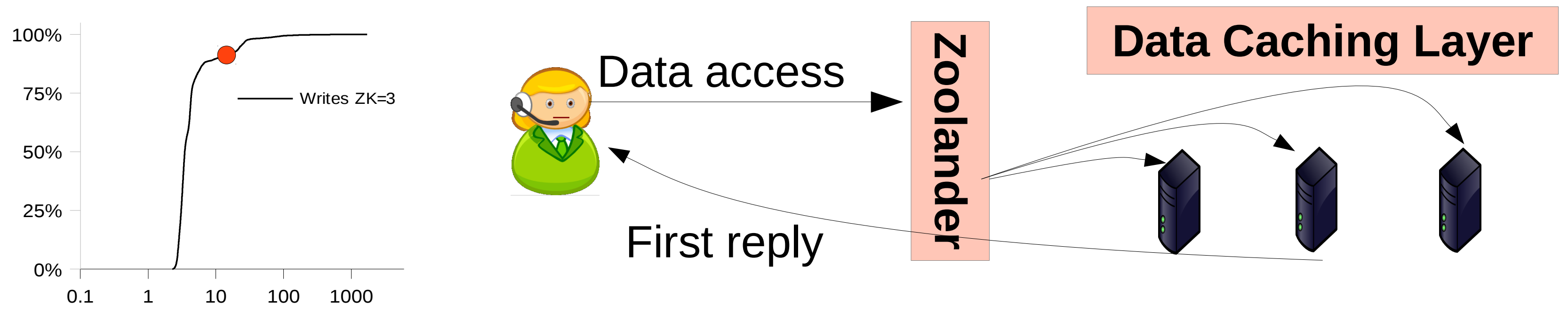


2 Zookeeper tests performed on different servers. Requests sent in the same order for each test

- Each point reflects a request's percentile in test #1 and #2
- Almost every quartile touched; statistical independence
- In-memory key value stores
  - Extremely fast; many OS operations can cause delays
  - Other workloads → Future work

Fat Tails in Key Values

1. Statistical Properties
2. **Core Model**
3. Model Validation



SLO: ?% of data accesses complete within 15ms(τ)

- **What is the probability that first reply exceeds 15ms?**

  (1 – Φ(15ms)) x (1 – Φ(15ms)) x (1 – Φ(15ms))

  Φ = Cumulative distribution function of access times

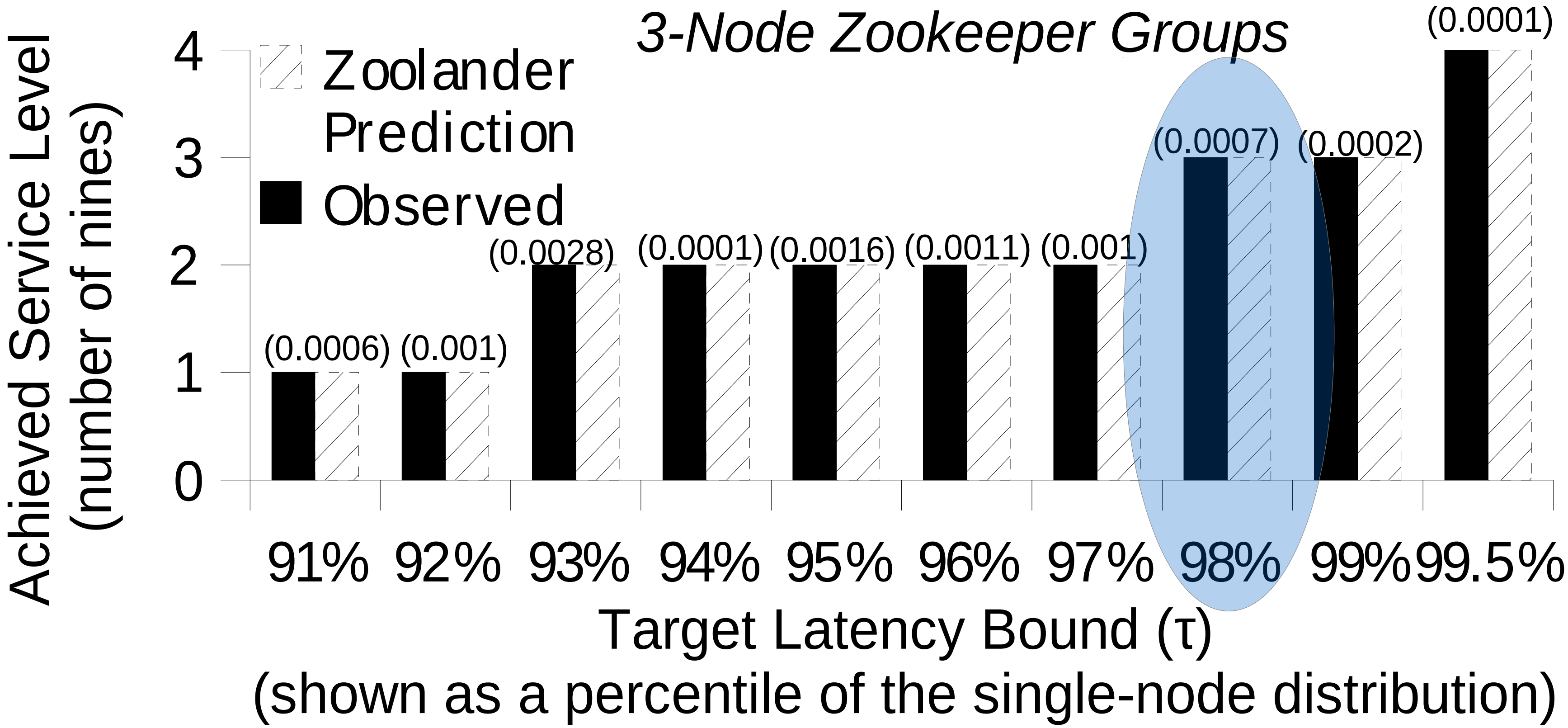
- **At scale (D), Service Level = 1 – (1 – Φ(τ))^{D}**

Fat Tails in Key Values

1. Statistical Properties
2. Core Model
3. **Model Validation**

Core model: Service Level = 1 – (1 – Φ(τ))^D

Test #1: Is the model accurate as τ varies?

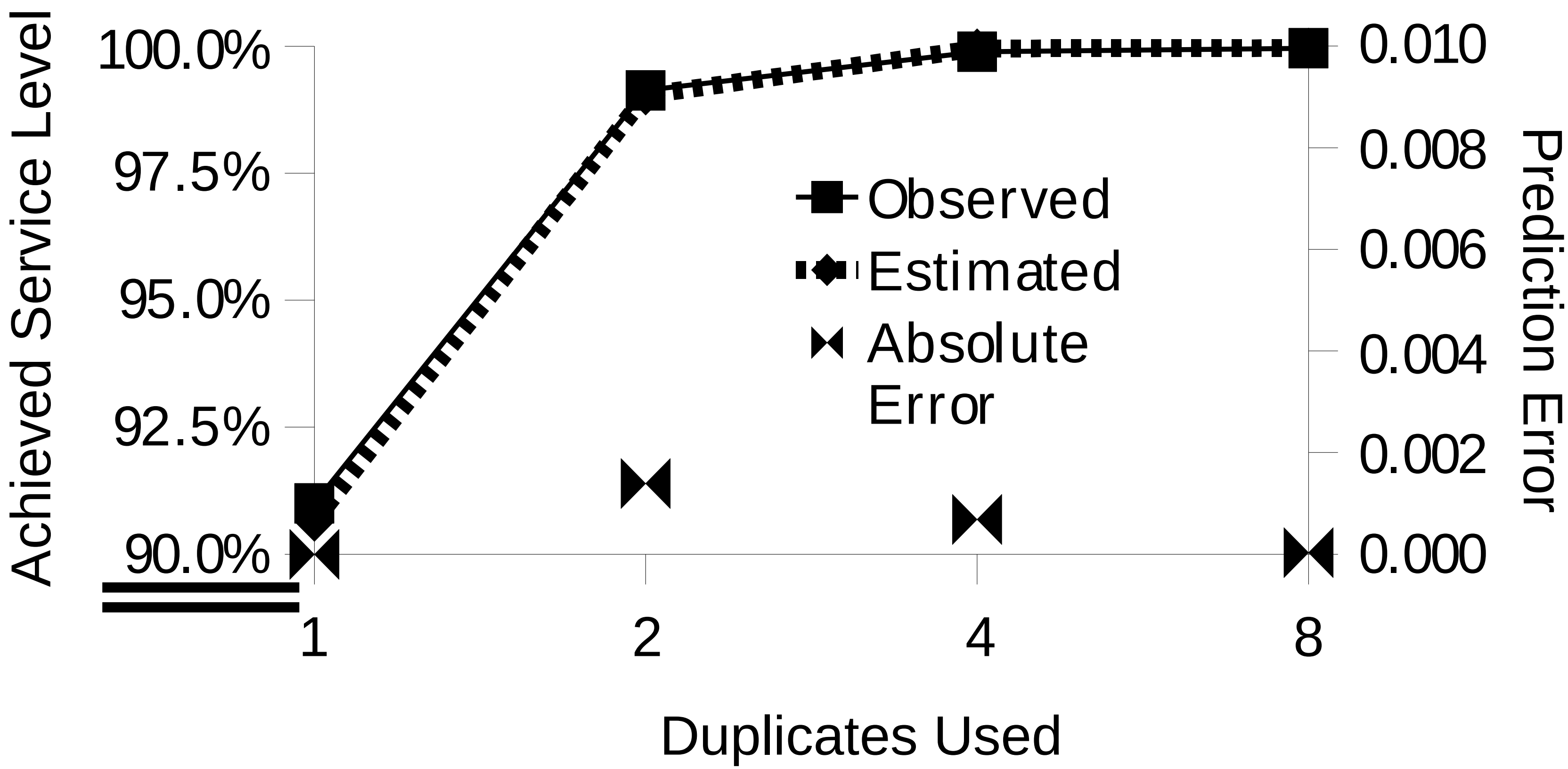


Fat Tails in Key Values

1. Statistical Properties
2. Core Model
3. **Model Validation**

Core model: Service Level = 1 – (1 – Φ(τ))^D

Test #2: Is the model accurate as D varies?



Meeting Strict SLOs Cost Effectively

1. Queuing
2. Model Driven Study
3. Zoolander at Scale

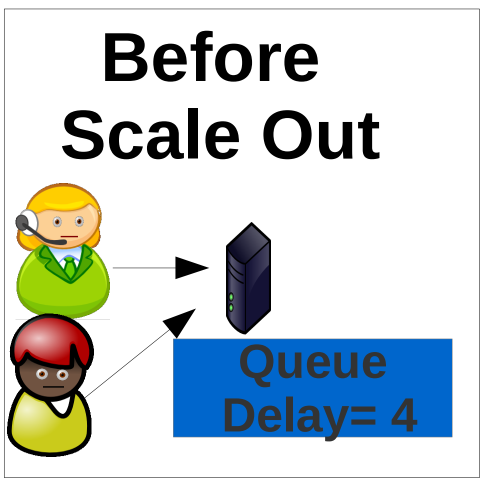
- Can we do it?
  - Study access-time tails in key value stores
  - Model replication for predictability on SLOs

- **Should we use replication for predictability to scale? Is it cost effective?**
  - Use our performance model to compare rep. for pred. against competing scale out approaches
  - Case study: Zoolander at scale

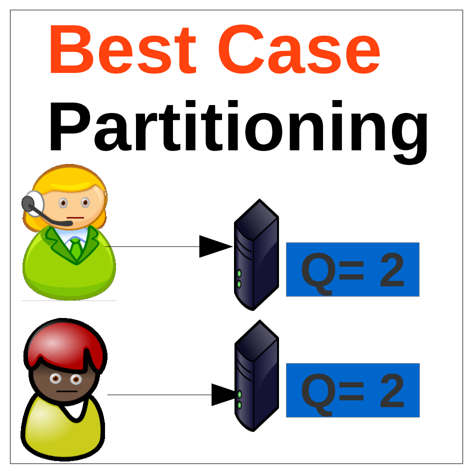
Meeting Strict SLOs Cost Effectively

1. Queuing
2. Model Driven Study
3. Zoolander at Scale

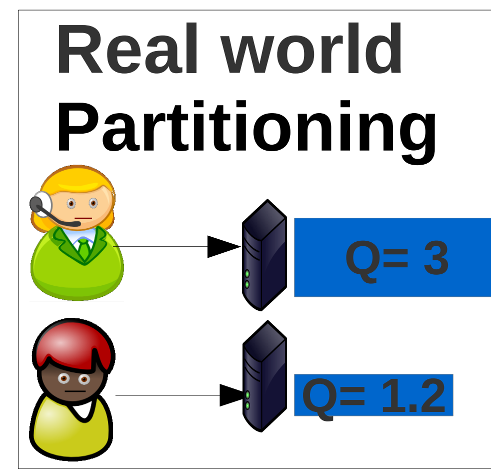
- Challenges: Duplicates share DC network and go through Zoolander
- Also, duplicates process requests at the same rate
  - Suffer the same queuing delay; Well modeled
  - Traditional scale out attacks queuing delay;“Divide the Work”



Only scale out via rep. for pred.



Captured by M/G/1



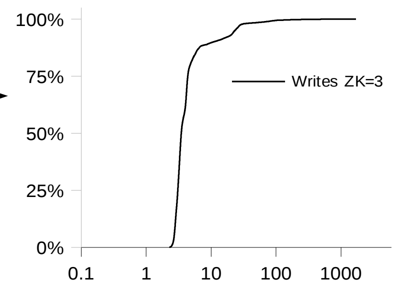
Hot spots, convoy, Consistency, etc.

Meeting Strict SLOs Cost Effectively

1. Queuing
2. **Model Driven Study**
3. Zoolander at Scale

- **Replication for predictability affects service times; traditional “divide the work” affects queuing delay**
  - When is replication for predictability definitely better?

$$queuing\ delay = F(arrival\ rate) \times service\ time$$



$$arrival\ rate = \frac{global\ arrival\ rate}{R}$$

R is number of replicas in traditional scale out

- **Post-queuing latency bound $\tau_{PQ} = \tau$ - queuing delay**

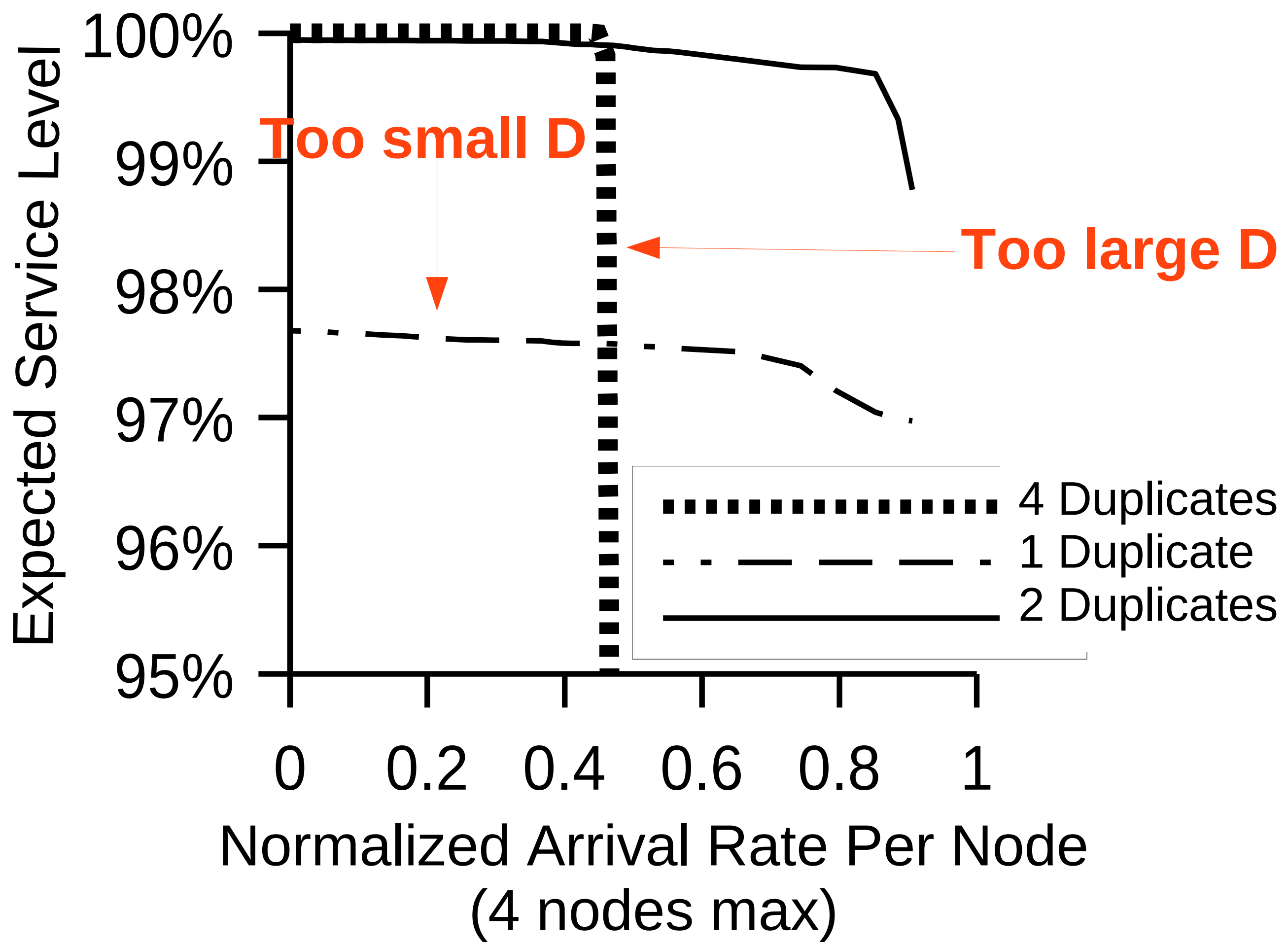
Meeting Strict SLOs Cost Effectively

Full model: Service Level = 1 – (1 – Φ(τ_{PQ}))^D

$$\text{Service Level} = 1 - \left(1 - \Phi\left(\tau - F\left(\frac{\text{global arrival rate}}{R}\right)\right)\right)^D$$

Meeting Strict SLOs Cost Effectively

1. Queuing
2. Model Driven Study
3. Zoolander at Scale



- *Does rep. for pred. strengthen SLOs?*
  Yes. Traditional scale out is limited by service time dist.

Best approach depends on arrival rate

Heavy arrivals per node = still a dumb idea

Moderate arrivals = Mixed strategy works well

Meeting Strict SLOs Cost Effectively

- **Should we use replication for predictability to scale? Is it cost effective?**
  - **Case study: Zoolander at scale**
  - **Zoolander is real middleware that currently works with Zookeeper, Cassandra, Redis, and memcached**
  - **TripAdvisor released details of its memcached [Gelfond '12]**
    - **We leased 144 EC2 units to test Zoolander under TripAdvisor's scale**

Meeting Strict SLOs Cost Effectively

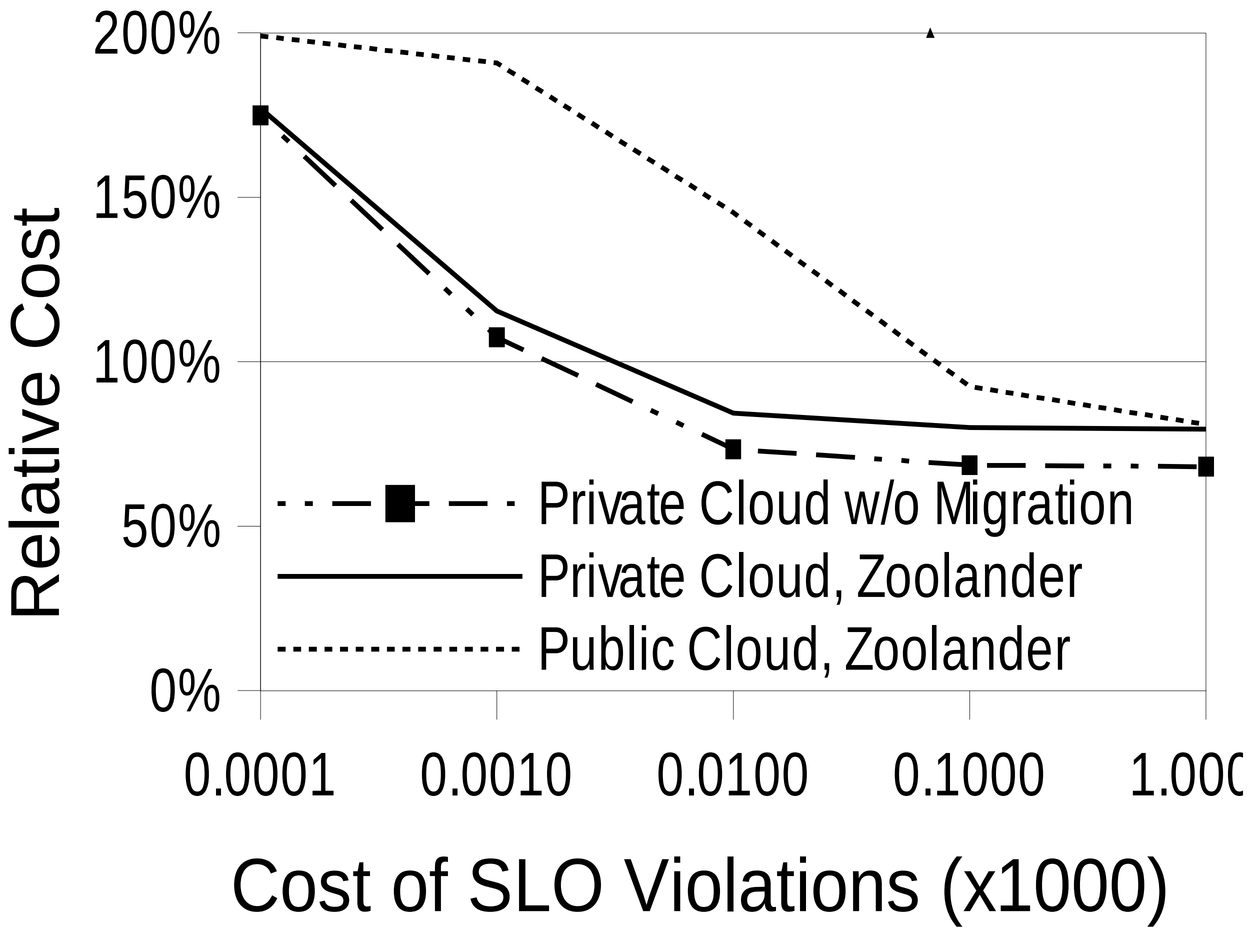
- Challenges:
  - Scale Zoolander to support 40M accesses per hour
  - Adapt Zoolander at night; accesses drop to 20M
  - **Strengthen SLO if possible—Be cost effective!**

- Competing, adaptive approaches
  - Make no changes at night
  - Turn off servers at night,
  - Replicate for predictability at night

Meeting Strict SLOs Cost Effectively

1. Queuing
2. Model Driven Study
3. **Zoolander at Scale**

Service Level Objective: Ensure 20 requests complete with 150ms



Zoolander reduced SLO violations by 32%!

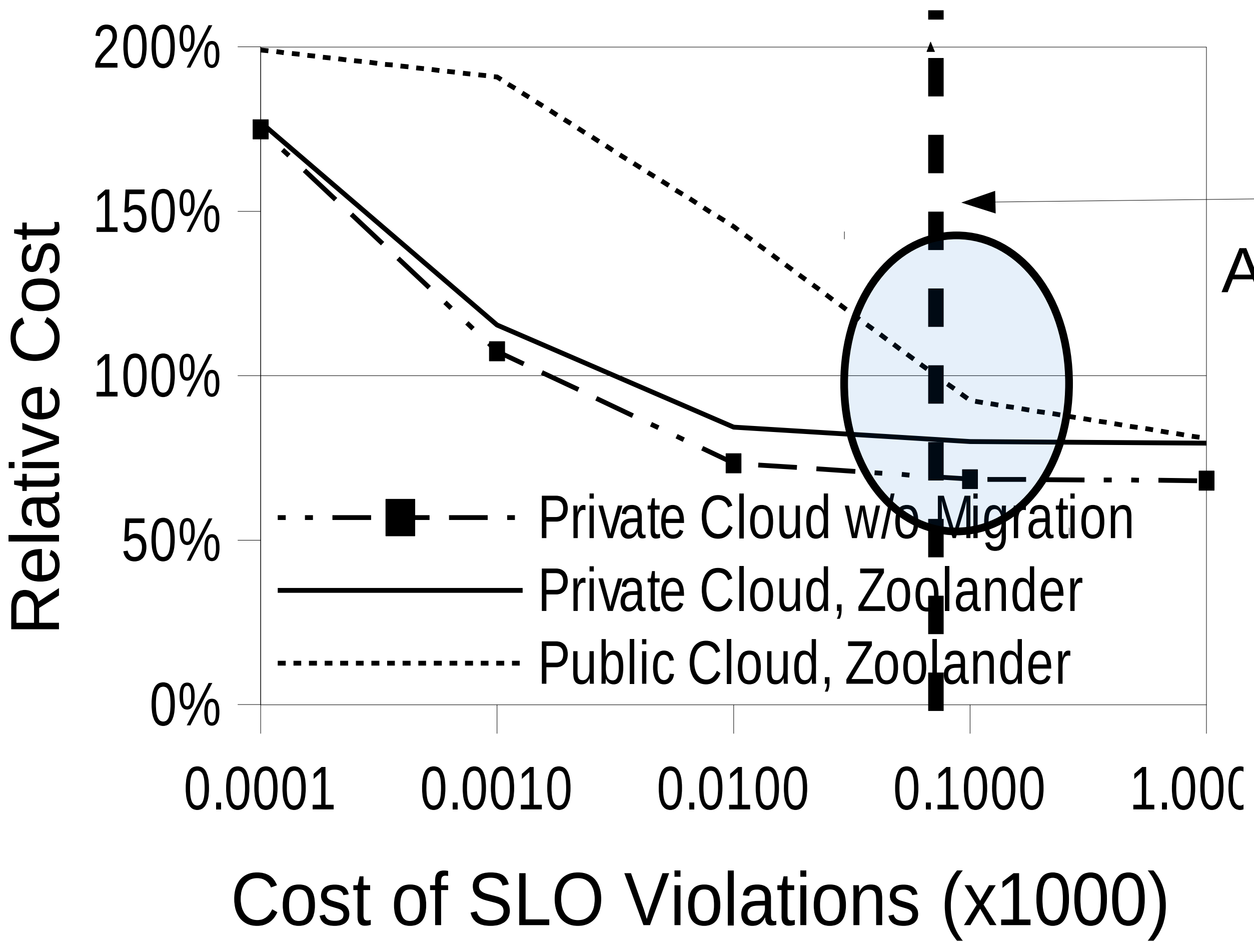
Zoolander is cost effective for private clouds

EC2 favors energy saving, save energy + hardware

Meeting Strict SLOs Cost Effectively

1. Queuing
2. Model Driven Study
3. Zoolander at Scale

Service Level Objective: Ensure 20 requests complete with 150ms



TripAdvisor
Ad revenue / Visitors * 1%
Cost = $0.068 per 1000

Meeting Strict SLOs Cost Effectively

1. Queuing
2. Model Driven Study
3. Zoolander at Scale

- **Fat tails are common, expected, and hard to remove in key-value stores**
- **Zoolander uses redundant execution to mask outlier access times and to meet SLOs cost effectively *at scale***
- **Traditional approaches *and* replication for predictability should be used for scale out. Analytic models can capture the benefits of both!**