PCAP Project: Characterizing and Adapting the Consistency-latency Tradeoff in Distributed Key-value Store

Muntasir Raihan Rahman

Joint work with
Lewis Tseng, Son Nguyen, Indranil Gupta, Nitin Vaidya

Distributed Protocols Research Group (DPRG)

http://dprg.cs.uiuc.edu

Key-value/NoSQL Storage Systems

- Key-value/NoSQL stores: $3.4B sector by 2018
- Distributed storage in the cloud
  - Netflix: video position (Cassandra)
  - Amazon: shopping cart (DynamoDB)
  - And many others
- NoSQL = “Not Only SQL”

Key-value/NoSQL Storage Systems (2)

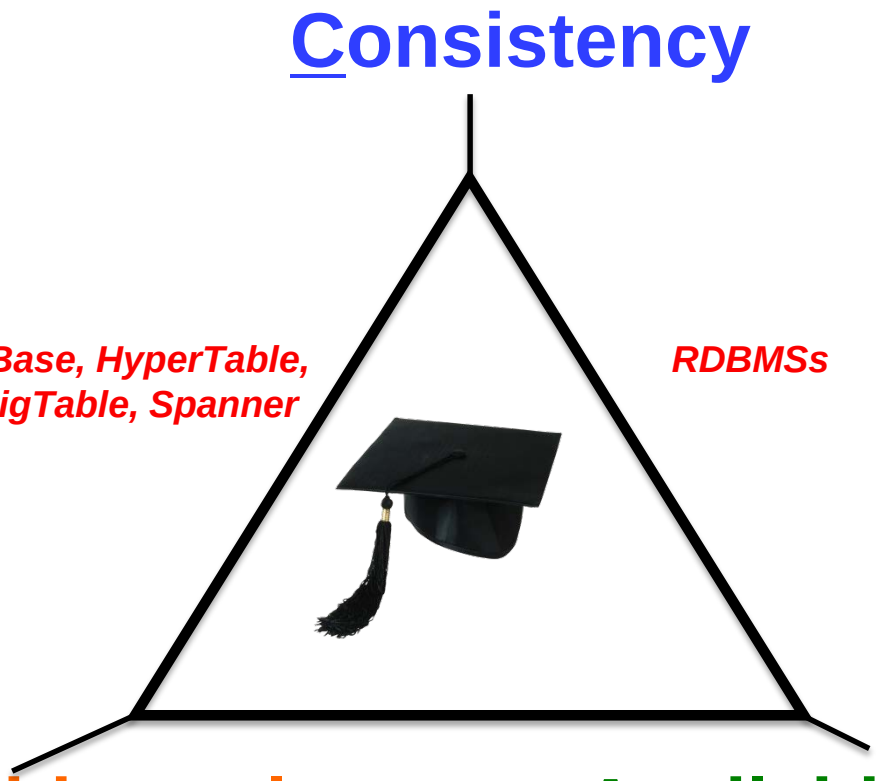
- Necessary API operations: get(key) and put(key, value)
  - And some extended operations, e.g., "CQL" in Cassandra key-value store
- Lots of open-source systems (startups)
  - Cassandra (Facebook)
  - Riak (Basho)
  - Voldemort (LinkedIn)
- Closed-source systems with papers
  - Dynamo

Key-value/NoSQL Storage: Fast and Fresh

- Cloud clients expect both
  - Availability: Low latency for all operations (reads/writes)
    - 500ms latency increase at Google.com costs 20% drop in revenue
    - each extra ms → $4 M revenue loss
  - Consistency: read returns value of one of latest writes
    - Freshness of data means accurate tracking and higher user satisfaction
    - Most KV stores only offer weak consistency (Eventual consistency)
    - Eventual consistency = *if writes stop, all replicas converge, eventually*
    - **Why eventual? Why so weak?**

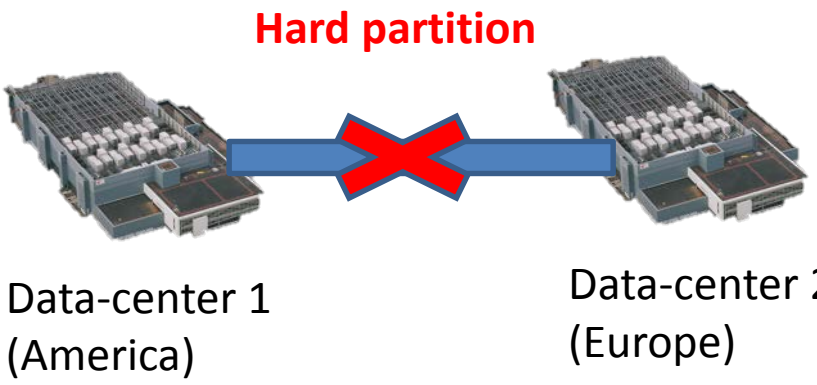
CAP Theorem → NoSQL Revolution

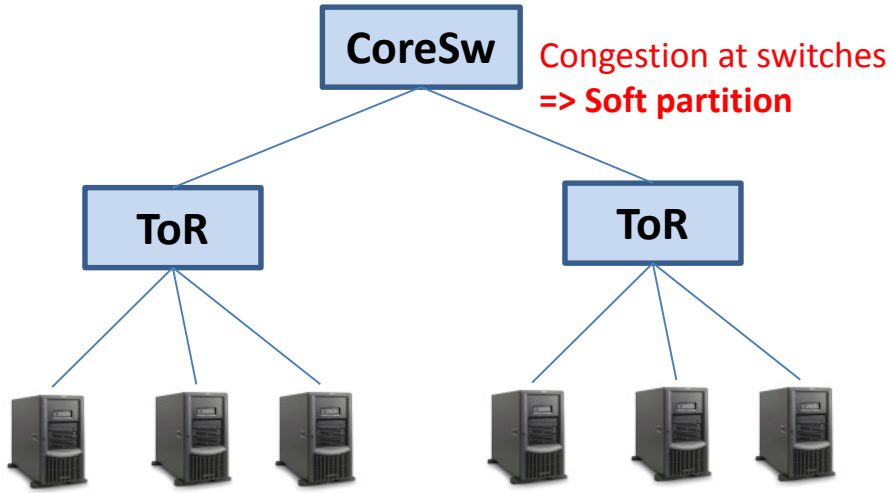
- Conjectured: [Brewer 00]
- Proved: [Gilbert Lynch 02]
- When network partitioned, system must choose either strong consistency or availability.
- Kicked off NoSQL revolution
- Abadi PACELC
  - If P, choose A or C
  - Else, choose L (latency) or C



Hard vs. Soft Partitions

- CAP Theorem looks at hard partitions
- However, soft partitions may happen inside a data-center
  - Periods of elevated message delays
  - Periods of elevated loss rates



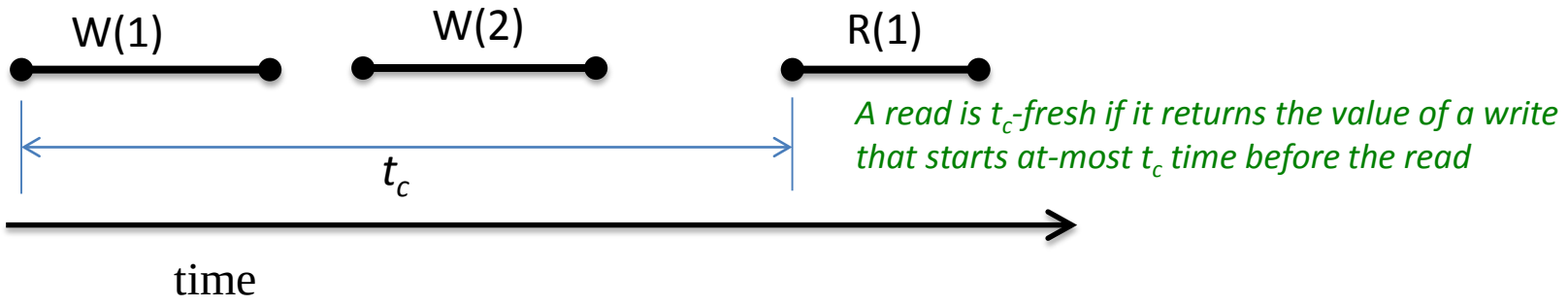


Our work:
From Impossibility to Possibility

- C → Probabilistic C (Consistency)
- A → Probabilistic A (Latency)
- P → Probabilistic P (Partition Model)

- Probabilistic Variant of CAP Theorem
- PCAP System to support SLAs (service level agreements) for single data-center
- GeoPCAP: extension to multiple geo-distributed data-centers

Probabilistic CAP



A read is t_c-fresh if it returns the value of a write that starts at-most t_c time before the read

p_{ic} is likelihood a read is NOT t_c-fresh

p_{ua} is likelihood a read DOES NOT return an answer within t_a time units

α is likelihood that a random host pair has message delay exceeding t_p time units

Probabilistic Consistency (p_{ic},t_c)

Probabilistic Latency (p_{ua},t_a)

Probabilistic Partition (α, t_p)

Bad network -> High (α, t_p)

To get better consistency -> lower (p_{ic},t_c)

To get better latency -> lower (p_{ua},t_a)

PCAP Theorem: ***Impossible to achieve both Probabilistic Consistency and Latency under Probabilistic Partitions***
if:
$t_c + t_a < t_p$ **and** $p_{ua} + p_{ic} < \alpha$

Towards Probabilistic SLAs

- Consistency SLA: Goal is to
  - **Meet** a desired freshness probability (given freshness interval)
  - **Maximize** probability that client receives operation's result within the timeout
  - Example: Google search application/Twitter search
    - Wants users to receive "recent" data as search
    - Only 10% results can be more than 5 min stale
      - SLA: $(p_{ic}, t_c)=(0.1, 5\ min)$
    - Minimize response time (fast response to query)
      - Minimize: p_{ua} (Given: t_a)

Towards Probabilistic SLAs (2)

- Latency SLA: Goal is to
  - **Meet** a desired probability that client receives operation's result within the timeout
  - **Maximize** freshness probability within given freshness interval
  - Example: Amazon shopping cart
    - Doesn't want to lose customers due to high latency
    - Only 10% operations can take longer than 300ms
      - SLA: (p_{ua}, t_a) = *(0.1, 300ms)*
    - Minimize staleness (don't want customers to lose items)
      - Minimize: p_{ic} (Given: t_c)

Meeting these SLAs: PCAP Systems

KV-store (Cassandra, Riak)

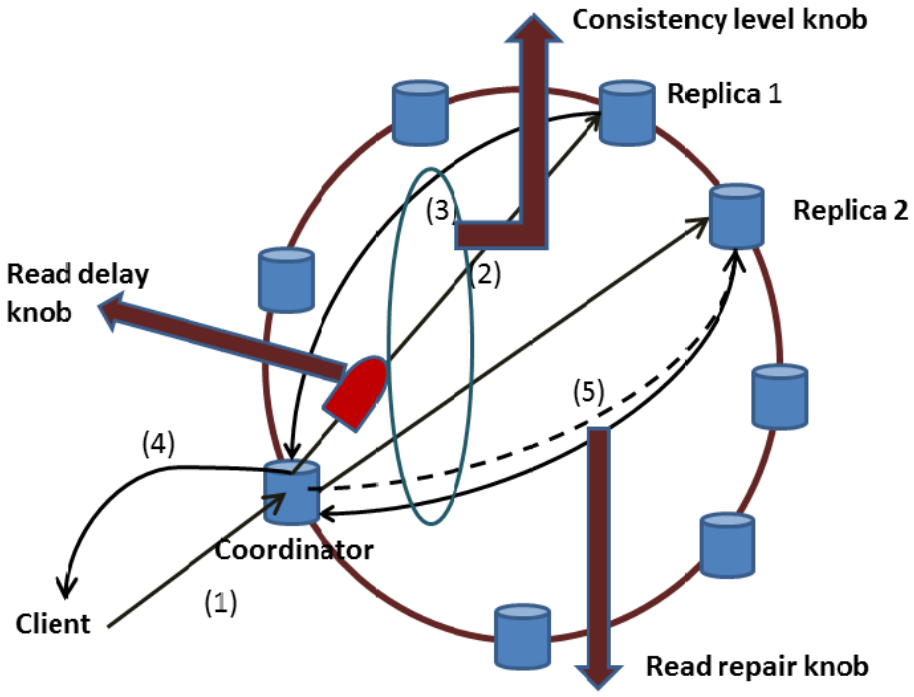
+ CONTROL KNOBS

+ ADAPTIVE CONTROL

= PCAP System

Single DC

Satisfies PCAP SLA



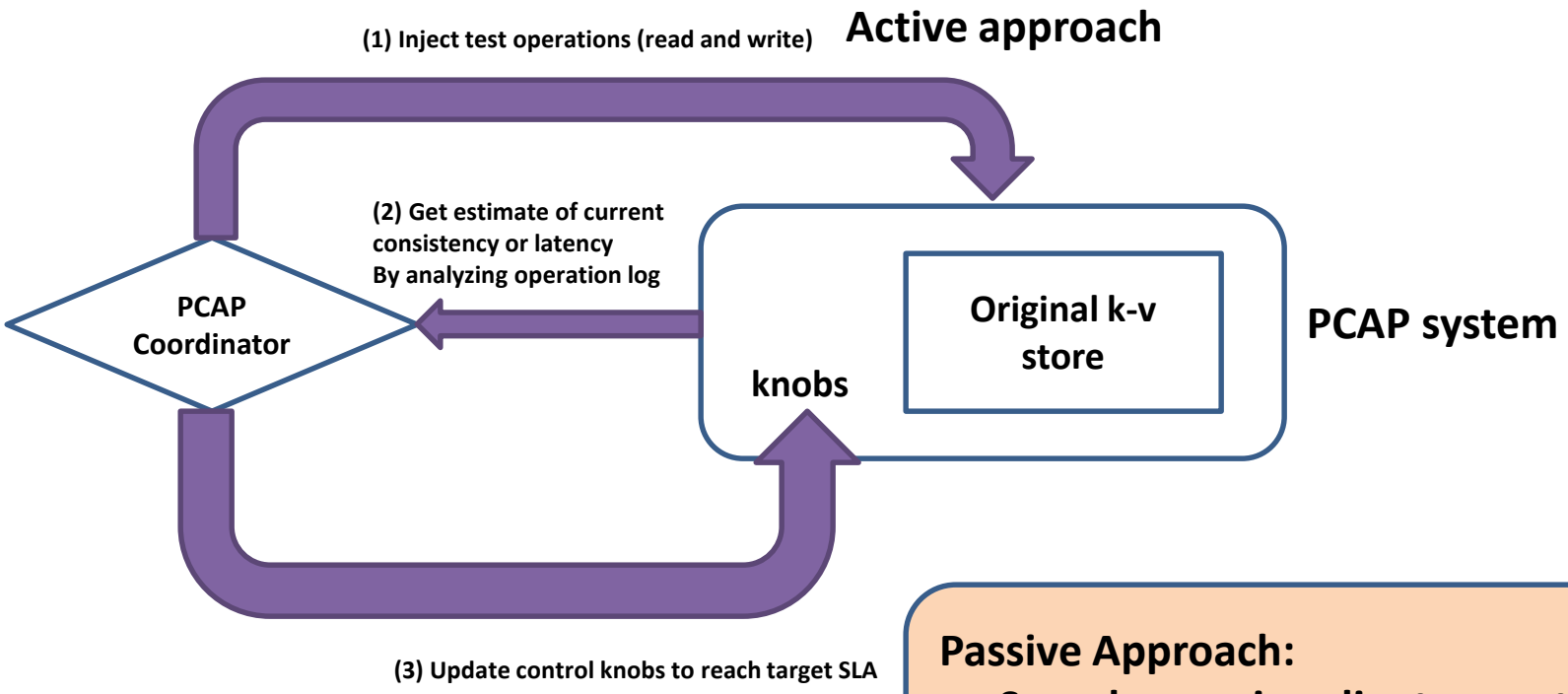
System assumptions:

- Client sends query to coordinator server which then forwards to replicas (answers reverse path)
- There exist background mechanisms to bring stale replicas up to date

Increased Knob	Latency	Consistency
Read Delay	Degrades	Improves
Read Repair Rate	Unaffected	Improves
Consistency Level	Degrades	Improves

Continuously adapt control knobs to always satisfy PCAP SLA

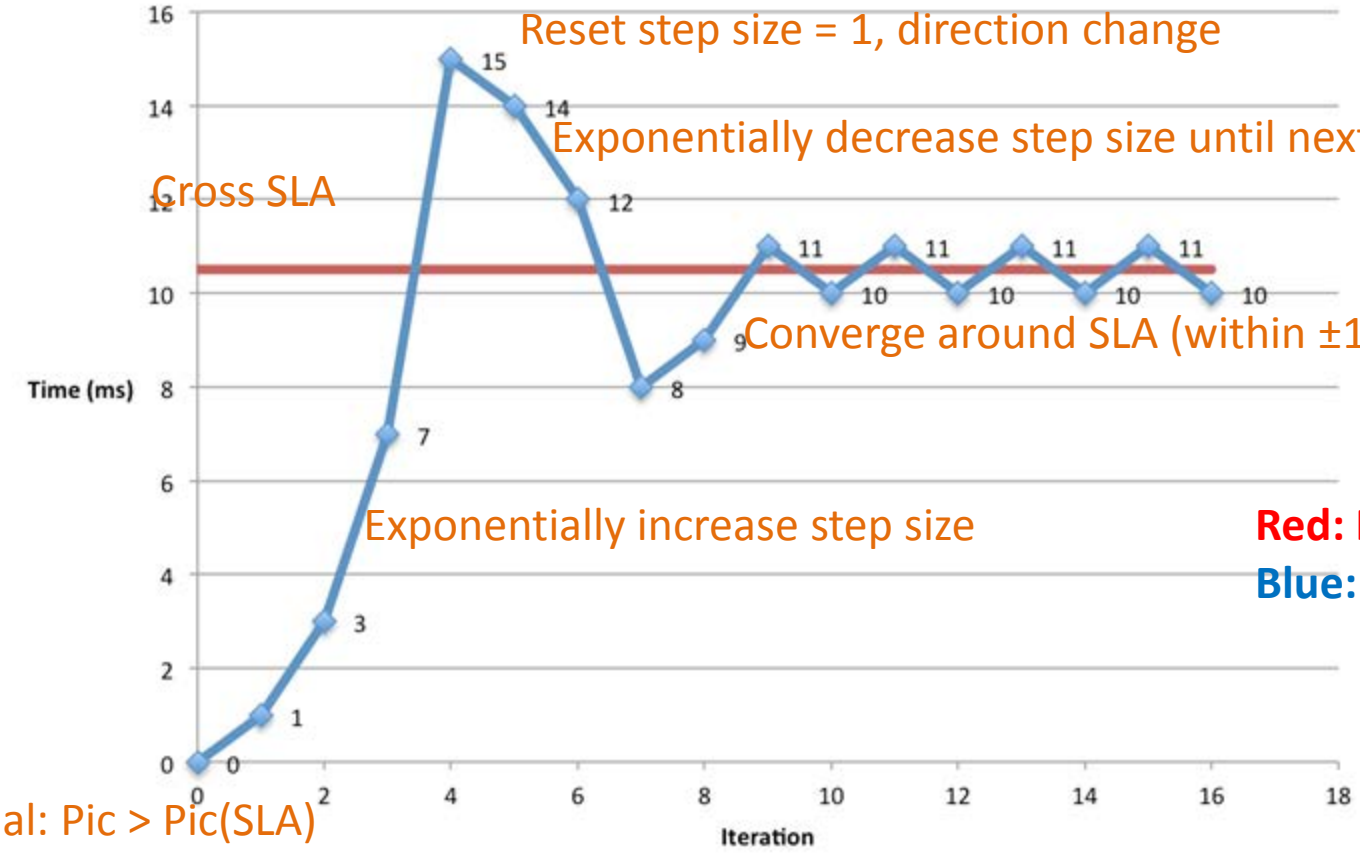
PCAP System Control Loop Architecture



Passive Approach:

- **Sample ongoing client operations**
- **Non-intrusive to client workloads**

Multiplicative Step Size Control



Red: Read delay (SLA)

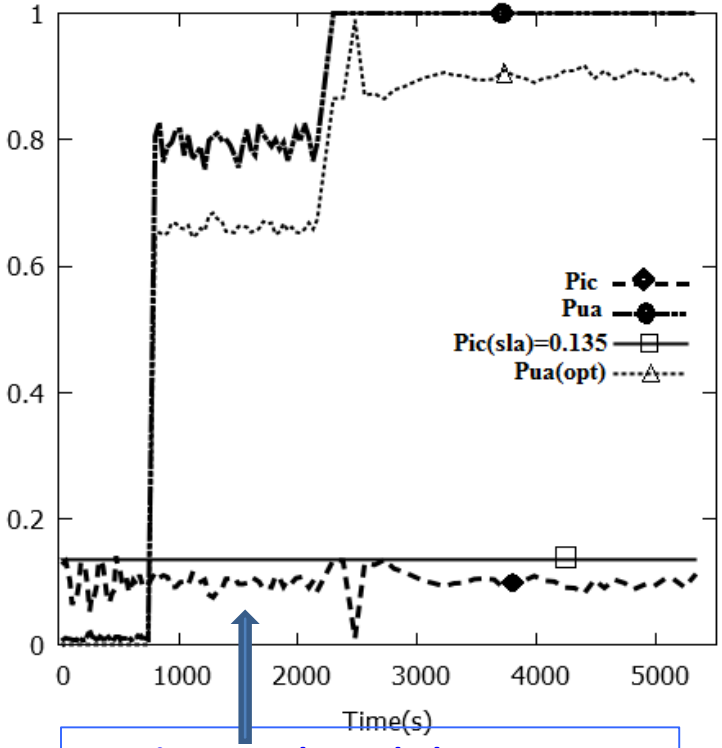
Blue: Read delay

Initial: Pic > Pic(SLA)

Faster convergence compared to unit step size change

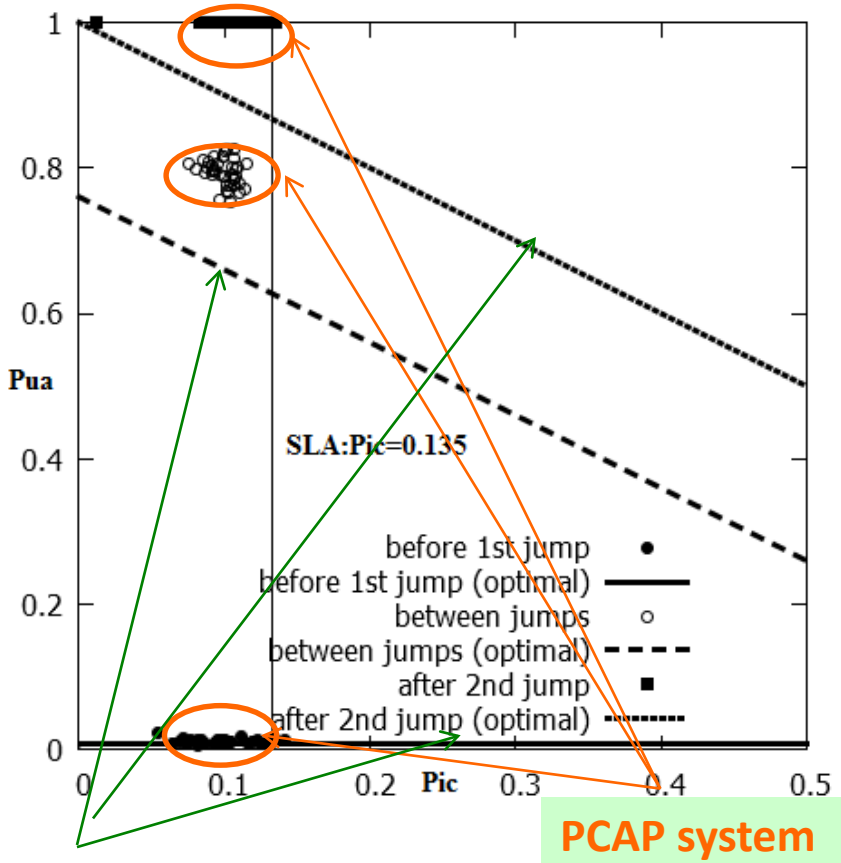
Meeting Consistency SLA for PCAP Cassandra (p_{ic}=0.135)

Mean latency =
3 ms | 4 ms | 5 ms



Consistency always below target SLA

Lognormal delay variation



Optimal envelopes under different Network conditions

PCAP system Satisfies SLA and close to optimal

Setup

- 9 server Emulab cluster: each server has 4 Xeon + 12 GB RAM
- 100 Mbps Ethernet
- YCSB workload (144 client threads)
- Network delay: Log-normal distribution

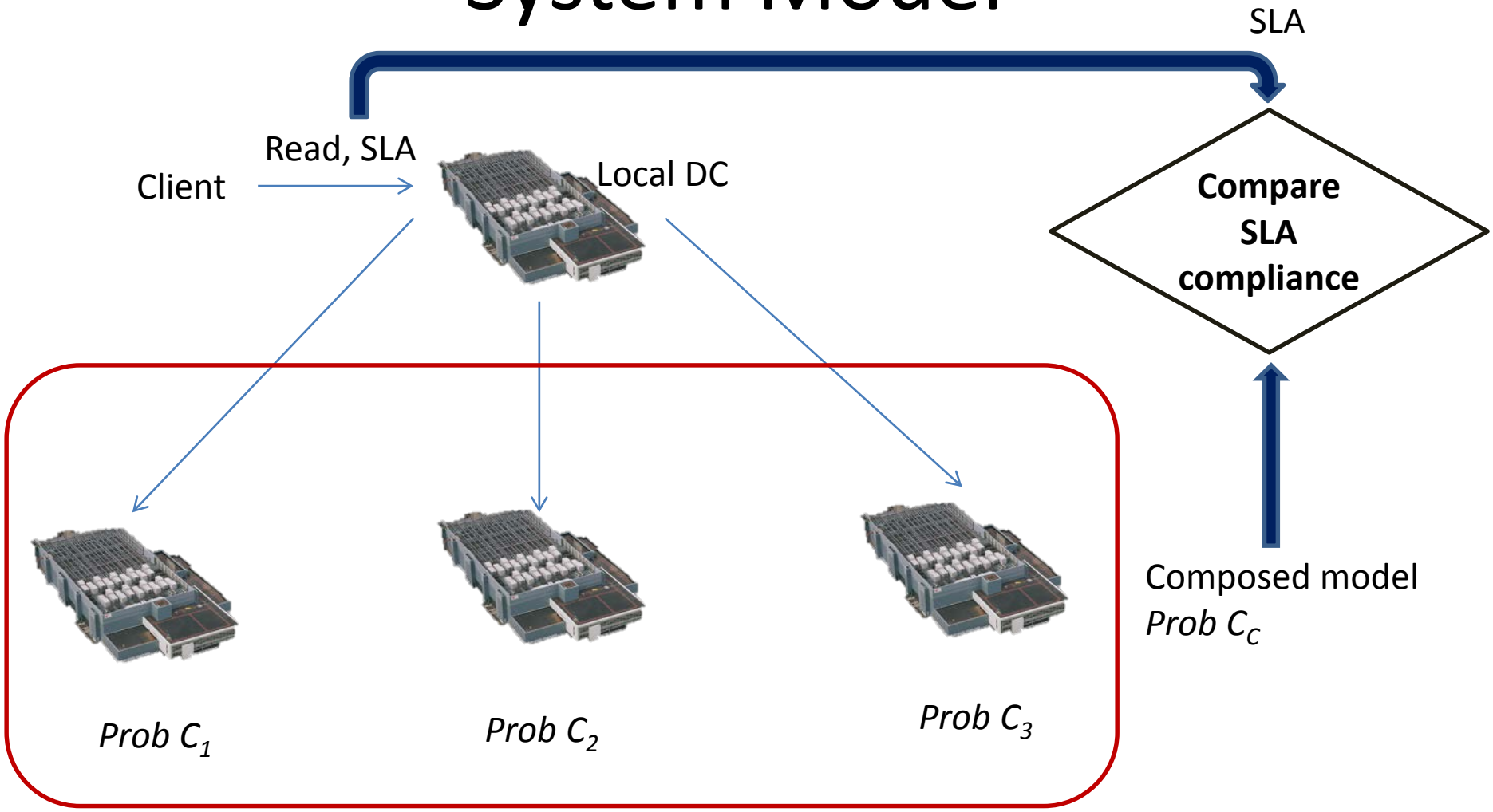
GeoPCAP

- Geo-distributed extension from single data-center to multiple geo-distributed data-centers
- Key techniques:
  - Probabilistic composition rules
  - New control algorithm
    - PCAP multiplicative controller oscillation

Composition Rules

- Assume each data-center is running single data-center PCAP system
- Each PCAP system has probabilistic consistency and latency models
- Composition rules allow us to characterize the global behavior across all data-centers
  - Enables us to check whether the composed system meets SLA
- Our rules generalize Apache Cassandra multi-data-center deployment rules

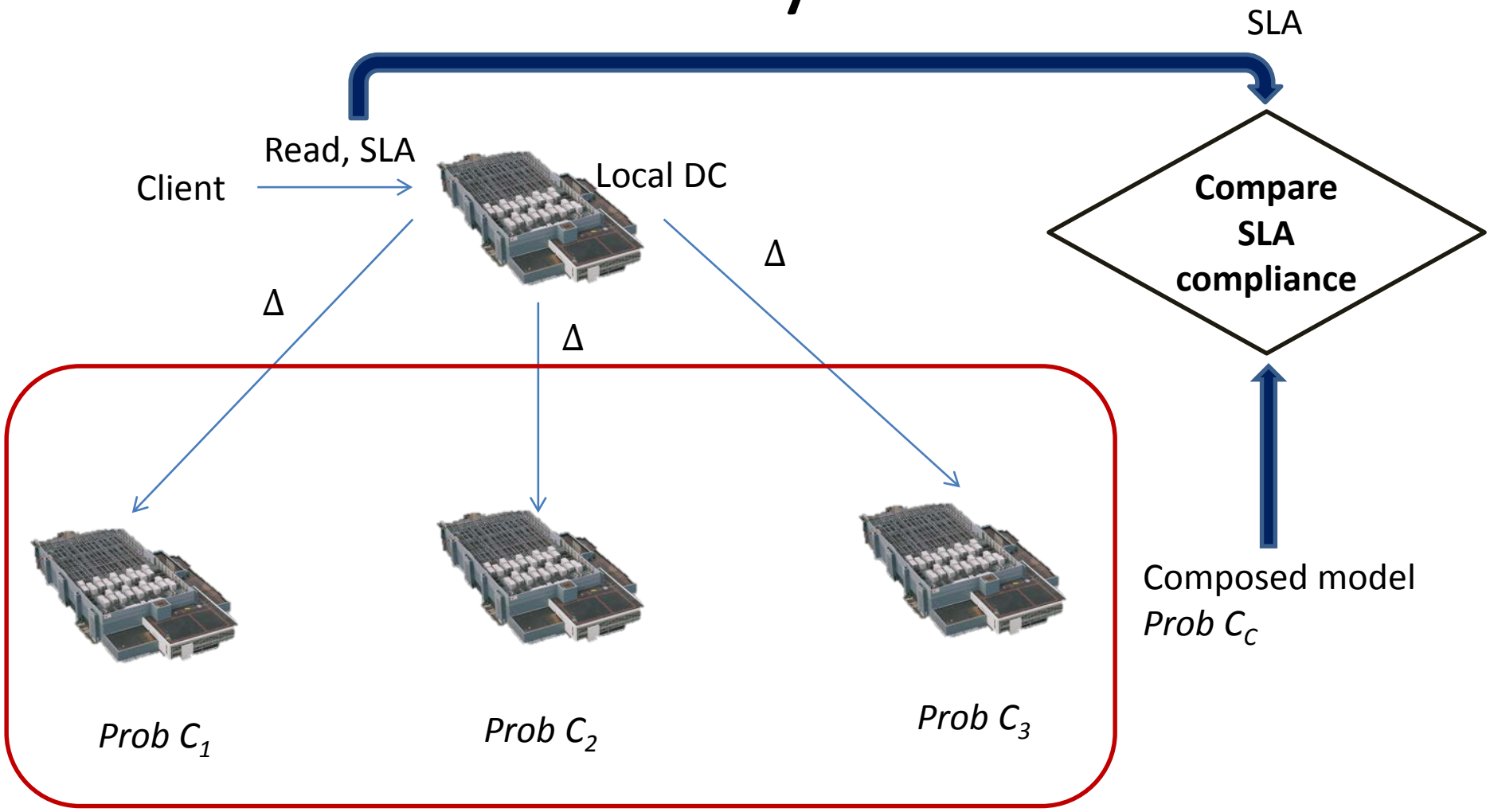
System Model



Given client C or L SLA:

- **QUICKEST: at-least one DC satisfies SLA**
- **ALL: each DC satisfies SLA**

Geo-delay Knob



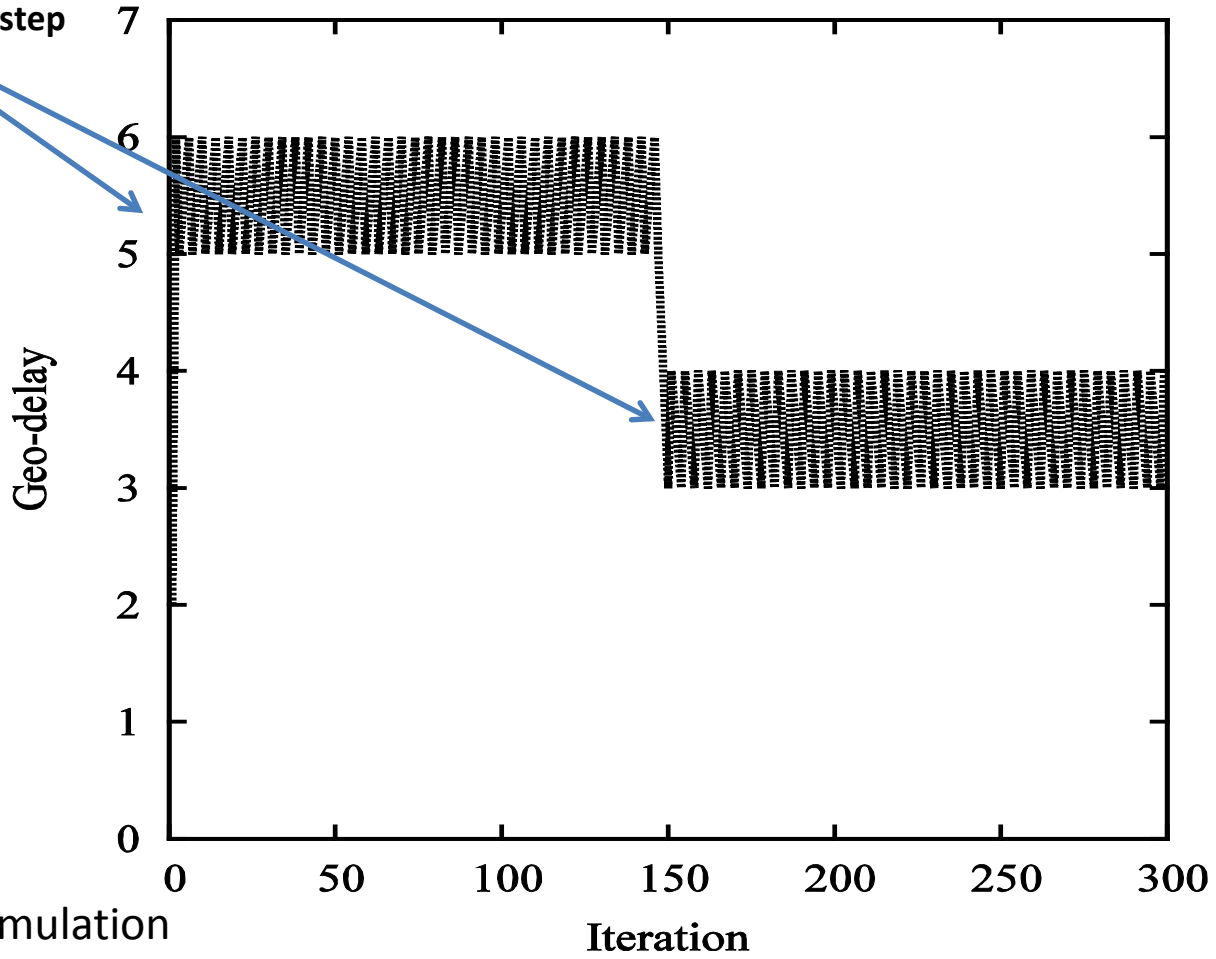
Δ: delay inserted at local DC before forwarding request to remote DC

Δ increase, wait longer at local DC, time for remote DC to sync, better consistency, worse latency

Δ decrease, vice versa

Limitation of Multiplicative Controller

At steady state, oscillate with unit step size change



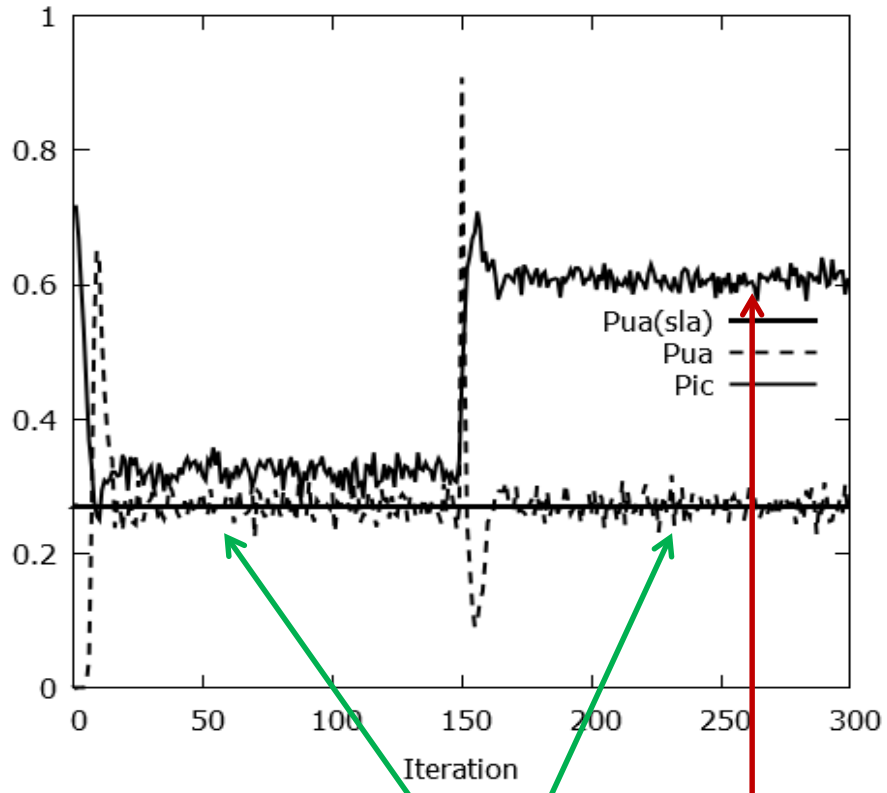
Monte-carlo simulation

- For each DC, use LinkedIN latency distribution to model (Prob C, Prob A)
- For WAN, use N(20,sqrt(2))

New Controller for GeoPCAP

- Based on PID control theory
- Error, *E = SLA – current metric*
- Knob change = $k_p \cdot E + k_d \cdot \frac{dE}{dt} + k_i \cdot \int E dt$
- (-) Overhead of tuning k_p, k_d, k_i to meet stability
- (+) But can reduce oscillations arbitrarily

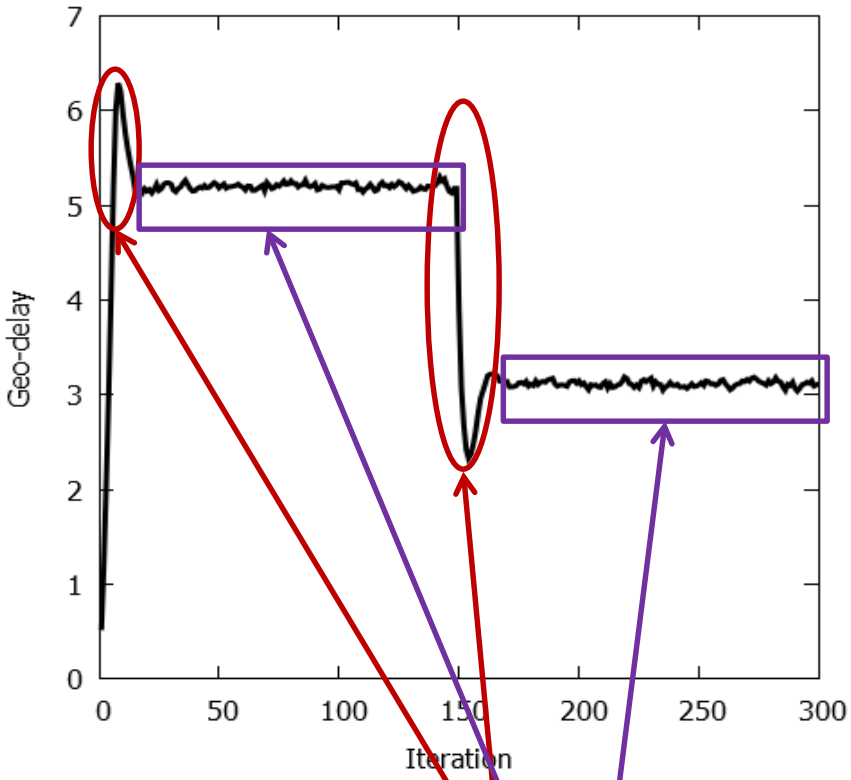
GeoPCAP L SLA (ALL)



N(20,sqrt(2)) | N(22,sqrt(2.2)

Latency SLA meet before and after jump

Consistency degrades after delay jump



N(20,sqrt(2)) | N(22,sqrt(2.2)

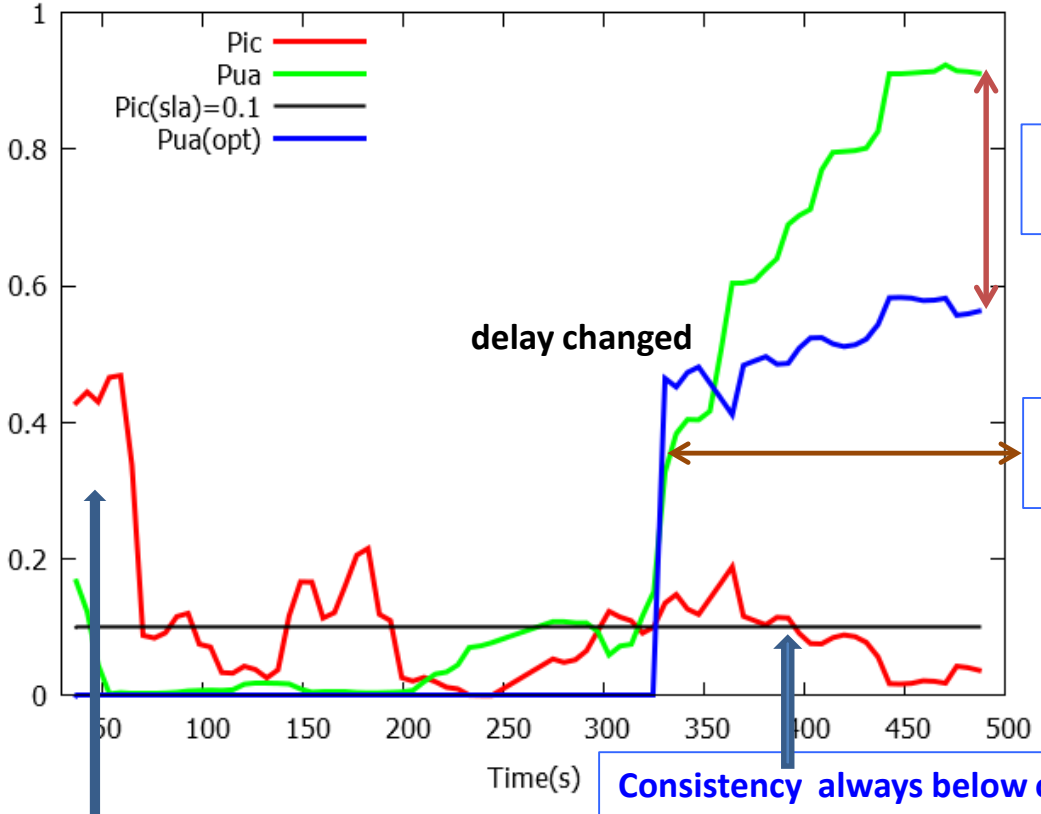
Fast convergence initially, and after delay jump

Reduced oscillation, compared to mult controller

Summary

- CAP Theorem motivated NoSQL Revolution
- But apps need freshness + fast responses
  - Under soft partition
- We proposed
  - Probabilistic models for C, A, P
  - Probabilistic variant of CAP theorem
  - PCAP system satisfies Latency/Consistency SLAs for single data-center
  - Integrated into Apache Cassandra and Riak KV stores
  - Extended to multiple geo-distributed data-centers

Meeting Consistency SLA for PCAP Cassandra (p_{ic}=0.2) [Passive]



Further from optimal, compared to active approach

Increased convergence time, compared to active approach

Consistency always below or near Target SLA

Initial conditions

P_{ic} and P_{ua} estimated from latest 100 operations from 5 servers

Related Works

	PCAP (our system)	Pileus (SOSP 2013)
SLA	Probabilistic Consistency/Latency SLA for single data-center	Family of Consistency/Latency SLA with utility values for geo-distributed settings
System Architecture	Writes can go to any server (P2P)	Writes only go to master replica (Master-slave replication)

	PCAP (our system)	PBS (VLDB 2012)
Consistency model (Freshness interval)	(W start time, R start time) models R/W overlap	(W finish time, R start time Cannot model R/W overlap

PBS (Probabilistically Bounded Staleness) metrics can be used in our PCAP system