Towards a Redundancy Aware Network Stack (RANS)

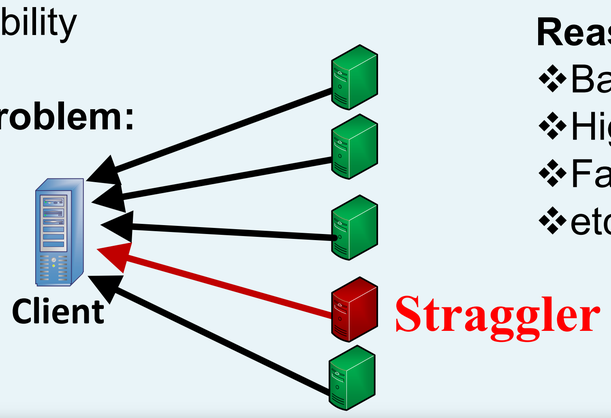
Ali Musa Iftikhar(Tufts), Fahad R. Dogar(Tufts), Ihsan A. Qazi (LUMS)

1. Motivation

High Performance needs of today's Datacenters:

- Predictable latency
- Fluid response times
- High availability

Straggler problem:



Reasons?

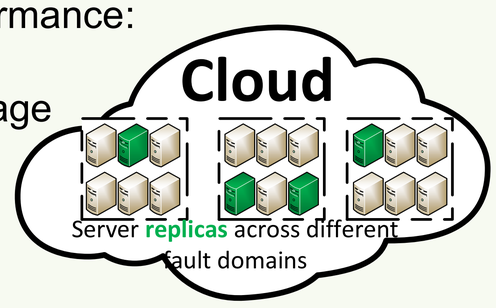
- Background tasks
- High load
- Failures
- etc

Stragglers hurt tail latency

2. Replication to the Rescue!

Replication techniques to improve performance:

- Cluster file systems
- Amazon S3, Windows Azure Storage
- Facebook's Haystackk



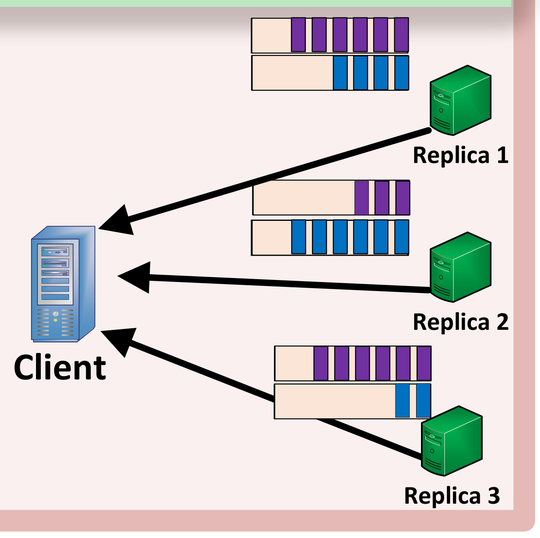
Current approaches

- Choose the best replica (**Difficult to predict** stragglers in advance)
- Adaptive replica selection (**Reactive, slow**)
- Initiate redundant requests – use first one that completes (beneficial only under low loads, **overloads the system** at higher loads)

3. RANS – Key Idea
Duplicate-aware scheduling

GOAL: Elevate redundant requests as a first class concept

- **Priority Queues**
- **Purging stale requests**
- **When is duplicate-aware scheduling useful?**



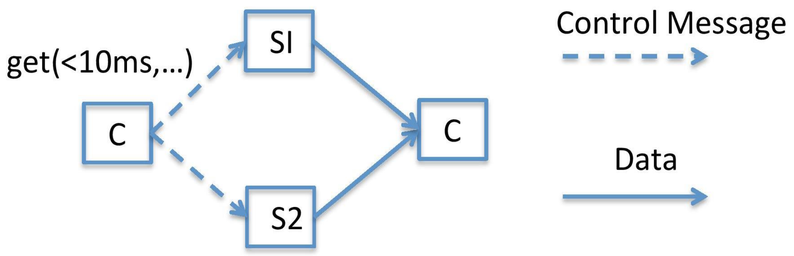
4. RANS - Stack

Layers	Role
App	Duplicate-Awareness
Interface	put/get; DAG-based
Transport	Point-to-Multipoint; Duplicate Flows; Purging
Network	Priority Queues; Purging
Link	Same as today's stack
PHY	Same as today's stack

5. Details of RANS layers

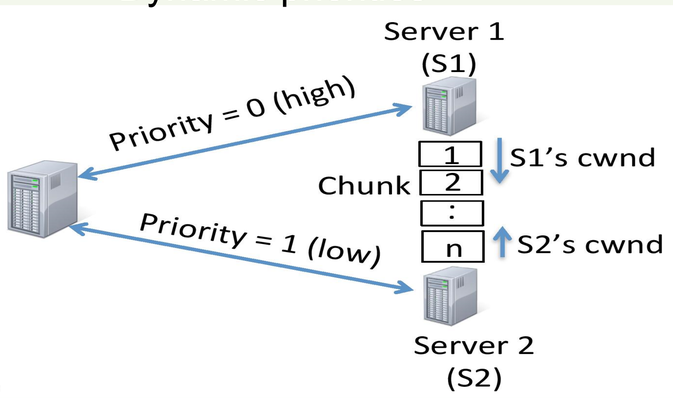
i. Interface

- Simple **DAG** for get().
- Client C reading from any of the two servers S1 and S2



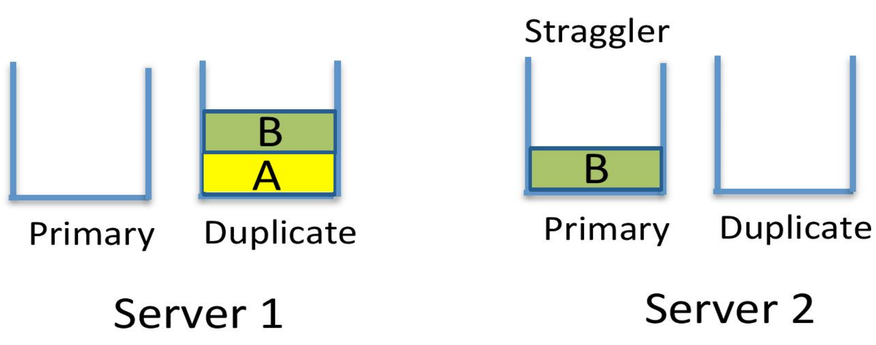
ii. RANS Transport

- Non-overlapping byte range
- Dynamic priorities

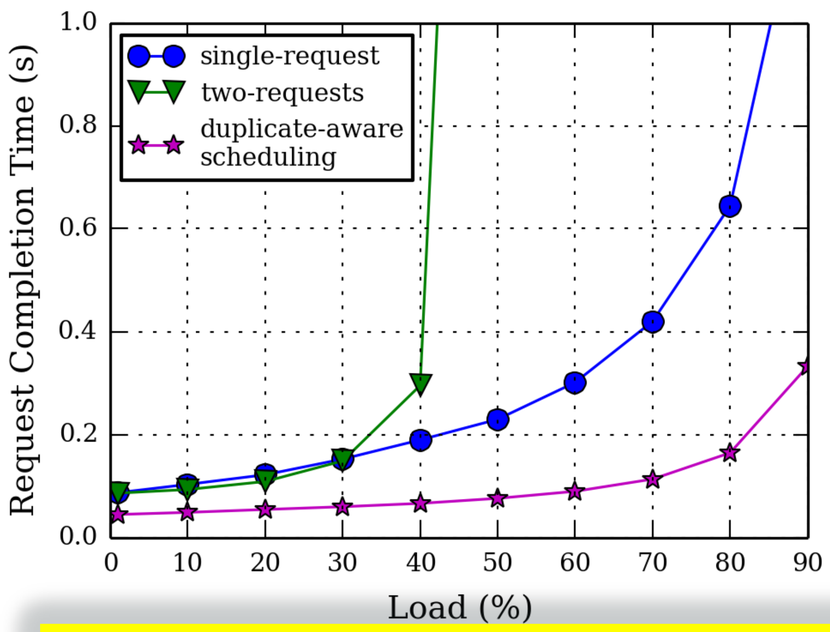


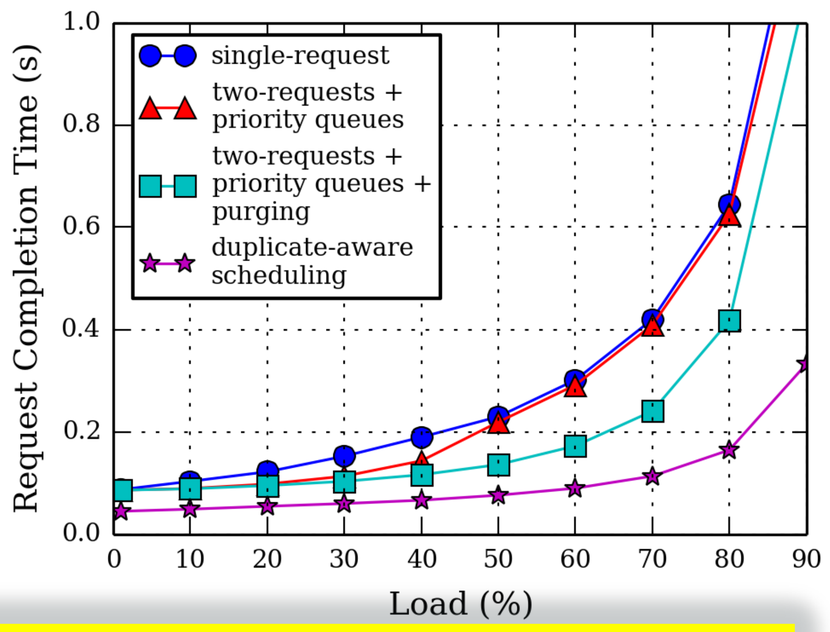
iii. Network Purging

- Purging strategies e.g:
  - weighted fair queuing
  - PFC



6. Initial results





RANS improves average request completion times at all loads

7. Ongoing work

- Designing a RANS transport protocol.
- Implementing duplicate-aware scheduling in HDFS and Cassandra.
- Evaluation on realistic test-beds and workloads