CSCI 2270: Advanced Topics in Database Management

Zephyr: Live Migration In Shared Nothing Databases For Elastic Cloud Platforms

"Cut Me Some Slack": Latency-Aware Live Migration For Databases

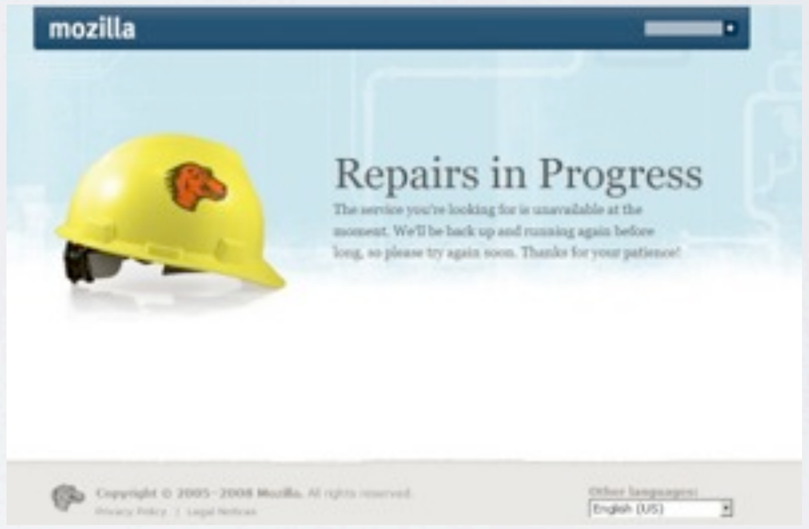
Yang Zou
yang@cs.brown.edu

BACKGROUND

- Infrastructures for large cloud platforms is challenged by applications that has **small data footprint** and **unpredictable load patterns**
- **System's operating cost** becomes critical if it's built on a **pay-per-use infrastructure**
- We want to minimize **cost** and guarantee **service** at the same time
- Elastic load balancing is wanted: 1)scale **up and down** based on the load 2) low cost to migrate data between hosts
- How can we achieve this ?

LIVE MIGRATION

- Why Live Migration?
- (Against **Stop & Copy**)



Repairs in Progress

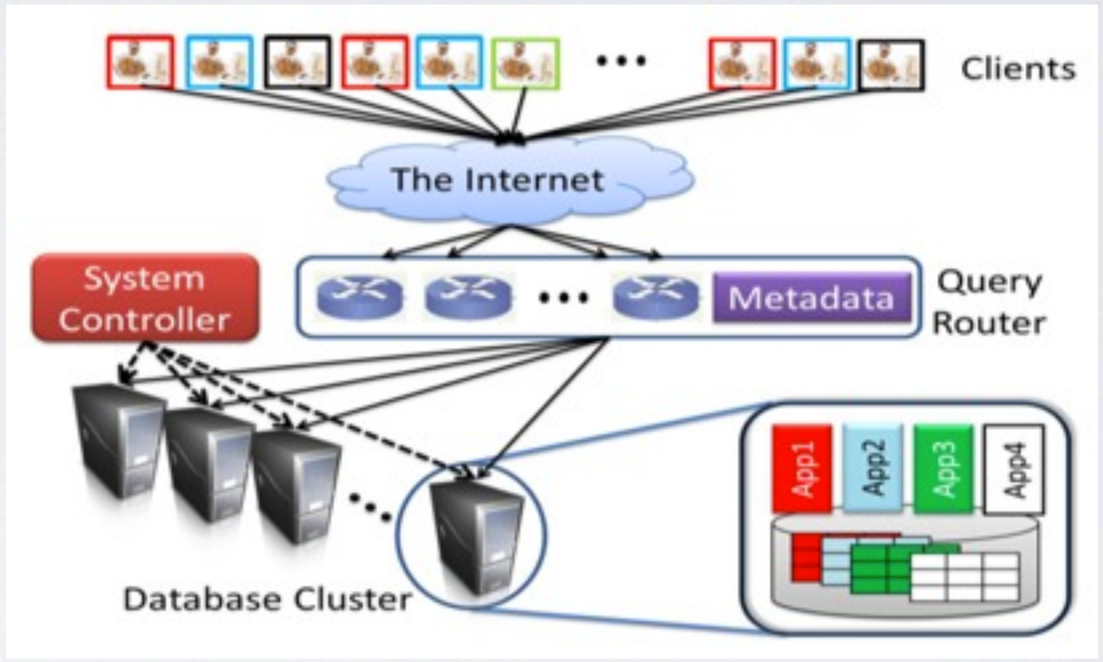
The service you're looking for is unavailable at the moment. We'll be back up and running again before long, so please try again soon. Thanks for your patience!

Copyright © 2005–2008 Mozilla. All rights reserved.
Privacy Policy | Legal Notices

Other languages:
English (US)

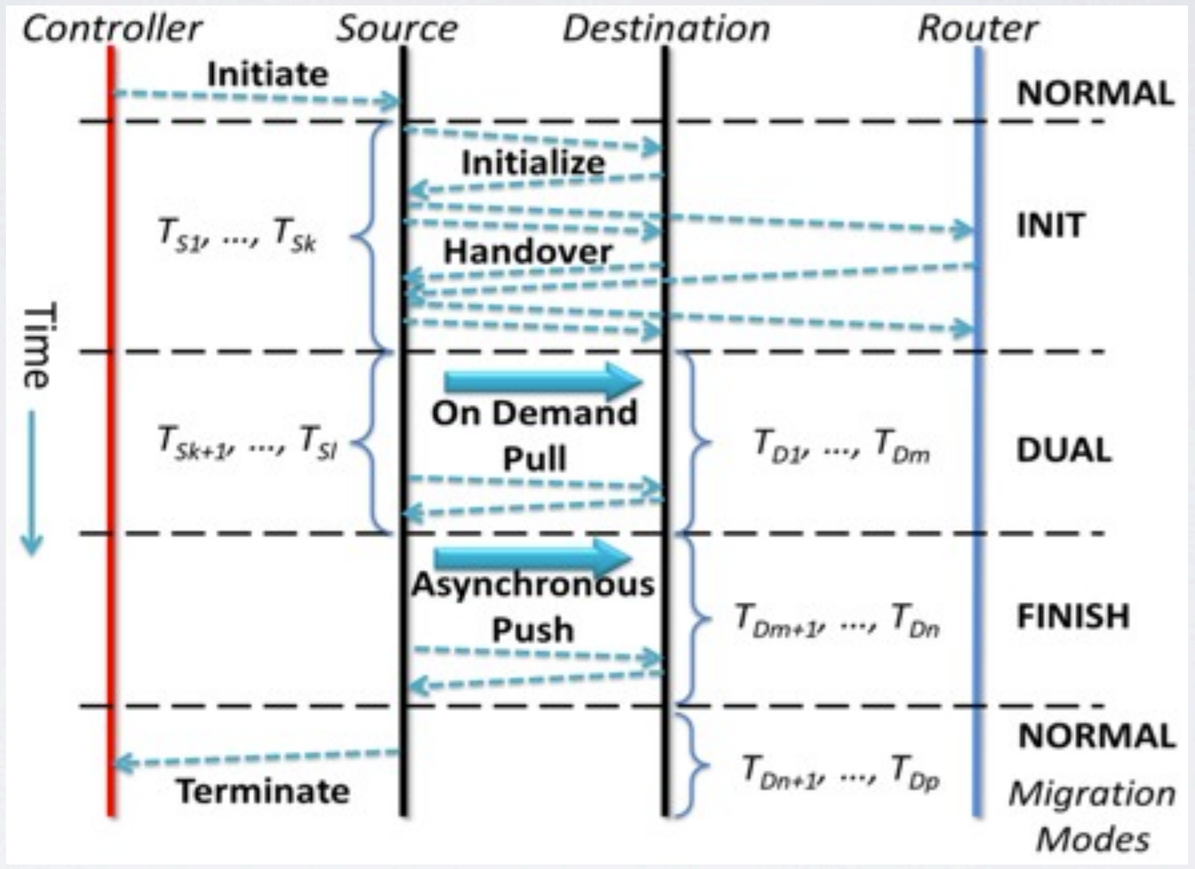
WHAT IS ZEPHYR

- Implemented in an open source RDBMS
- First complete end-to-end solution for live migration in a shared nothing database architecture
- Very light-weighted



HOW ZEPHYR WORKS

- Normal Mode
- Init Mode
- Dual Mode
- Finish Mode



KEY IDEAS

- Init Mode

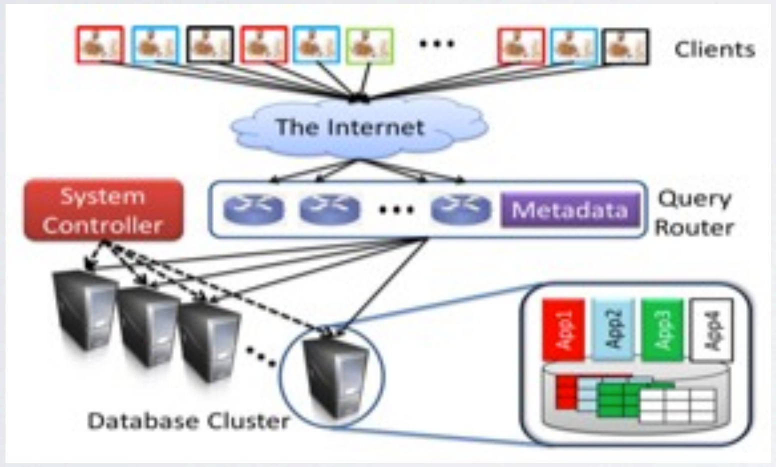
- Source node bootstraps destination node by sending wireframe (schema, data definitions, etc.)
- Source node is still the unique owner of Dm

- Dual Mode

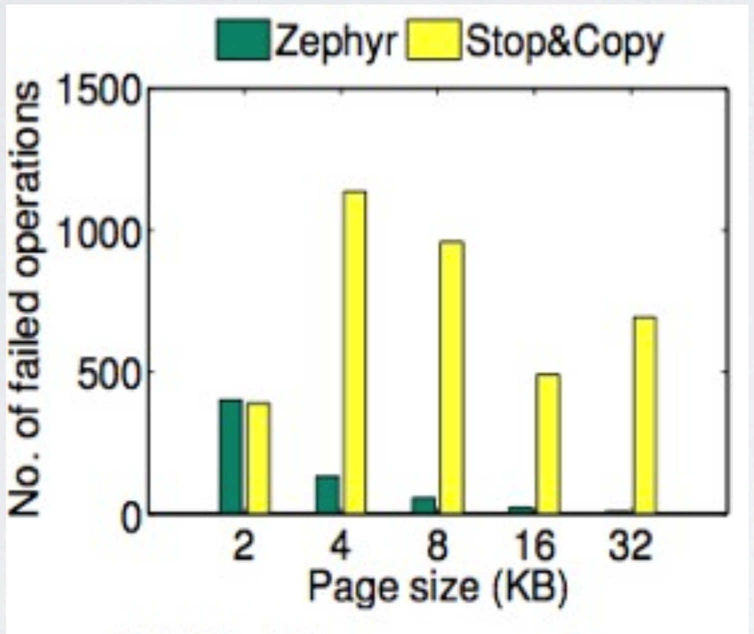
- Destination node notifies the source node about the completion of initialization
- Source node tells the query router to direct all new txns to destination node
- Both Source node and Destination node are the owner of Dm
- Pages are transferred to destination node on-demand
- Source node give up the ownership of Dm and destination owns Dm itself

- Finish Mode

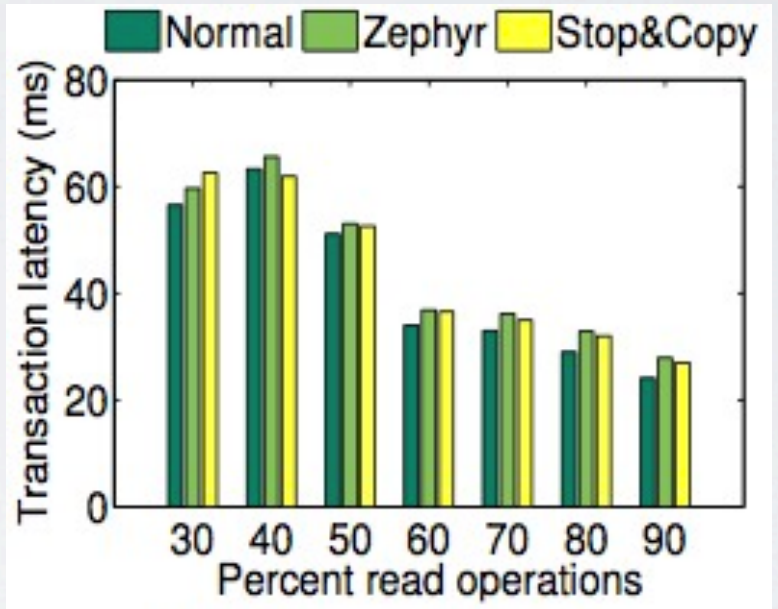
- Source node transfers the remaining pages of Dm to the destination node
- Source node initiates the termination of migration
- Source node and destination node work on normal mode



EXPERIMENTAL RESULTS



(b) Database page size

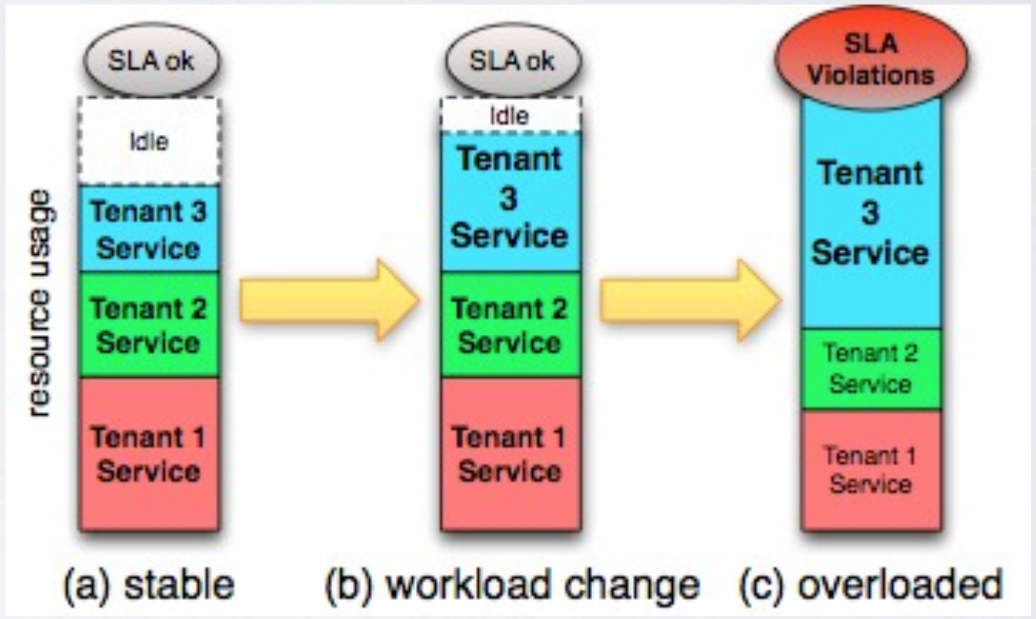


(c) Average latency

ANY QUESTIONS?

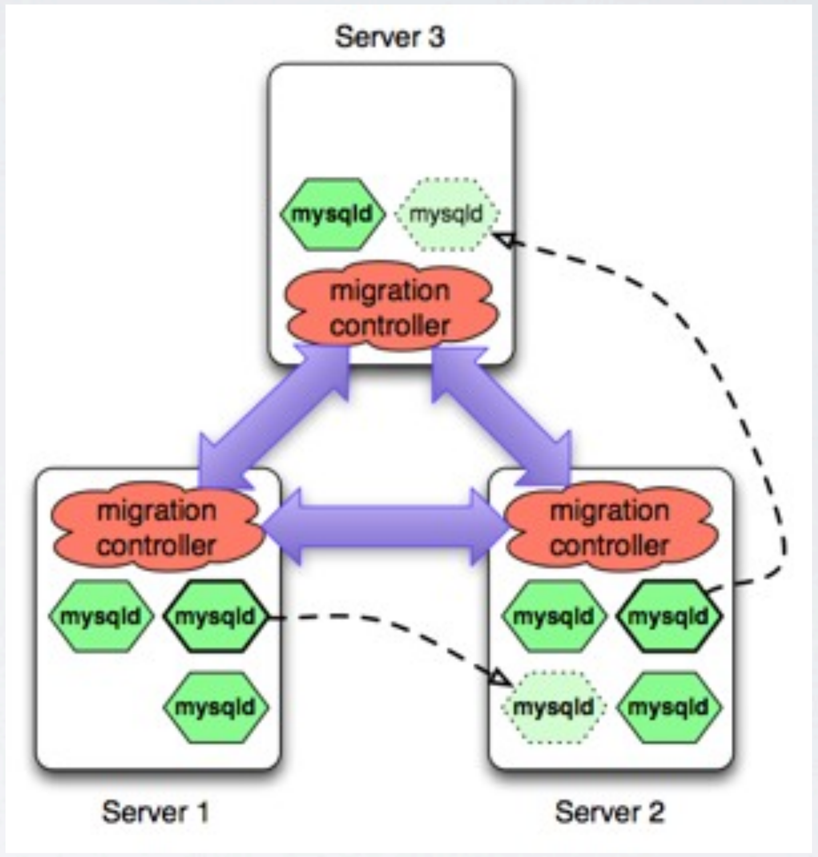
"Cut Me Some Slack": Latency-Aware Live Migration For Databases

- "Shared something database"
- Migrating data elegantly
- Can be implemented outside of a database product
- Used several existing tools, like XtraBackup, pv



SLACKER KEY IDEAS

- Slacker Architecture
  - Each server runs an instance of Slacker
  - Slackers migrates MySQL instances between servers that run Slacker

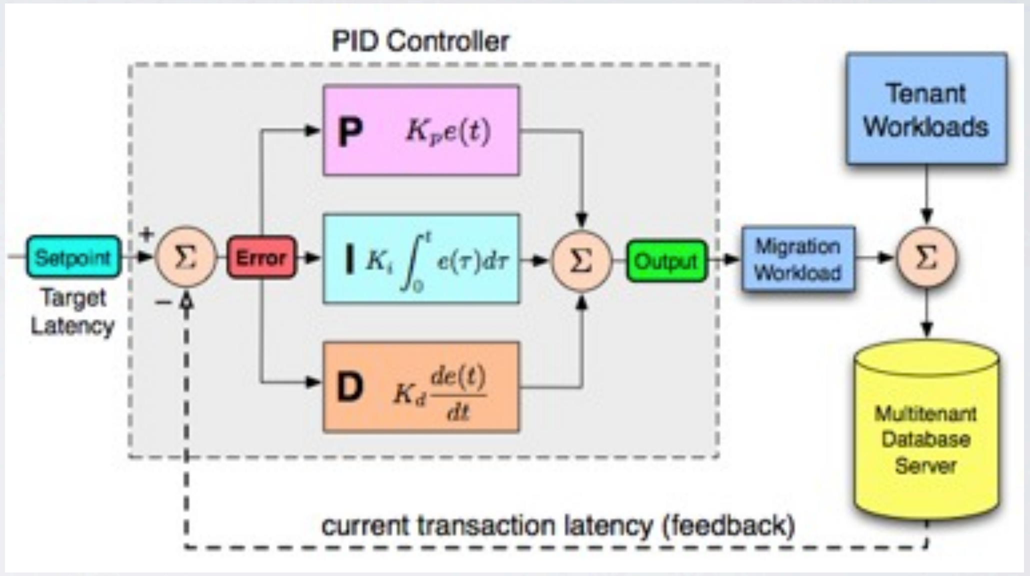


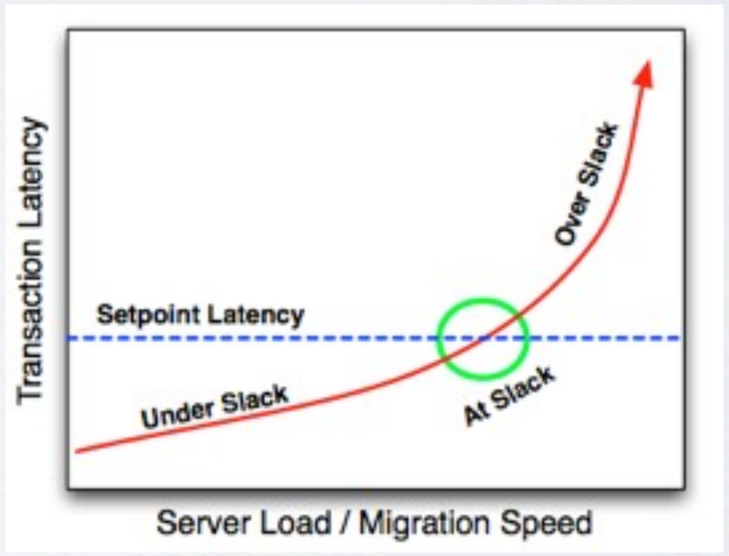
SLACKER KEY IDEAS

- Migration Slack & Setpoint Latency
    - Resources can be used for migration
    - The latency that maintains acceptable query performance
    - Migration throttling: control the cost of each migration
    - Need to adjust the cost on-the-fly (based on workloads)

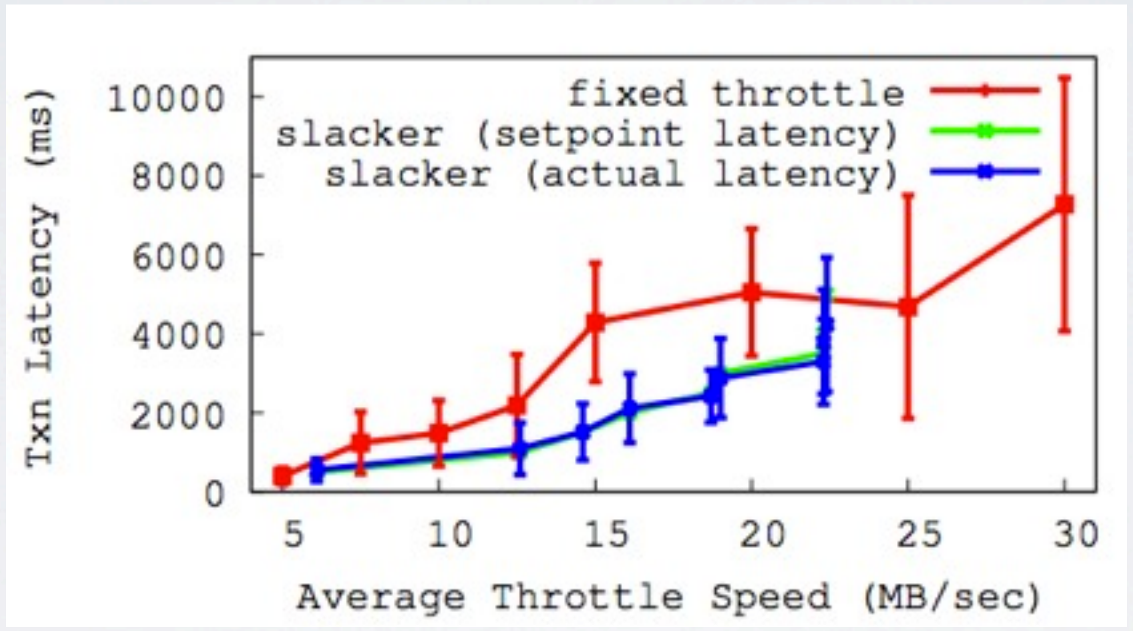
SLACKER KEY IDEAS

- Adaptive Dynamic Throttling
    - Determine the speed of migration according to the slack
    - Adjust the speed of migration according to the slack in real time
    - Speed of migration is controlled by PID
        - Control the migration speed to make the transaction latency as close as the setpoint latency





EXPERIMENTAL RESULTS



CONCLUSION

- Zephyr: how to do migration
- Slacker: how to migrate data as fast as possible
- Zephyr + Slacker = Live Migration in H-Store (Hopefully...)

ANY QUESTIONS?

THANKS!