Deterministic DBMS and Future

Yu-Shan Lin

5th August 2020

Outline

- Goal for Cloud RDBMSs
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

Outline

- Goal for Cloud RDBMSs
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

Definition

- A ***cloud DBMS*** is a DBMS designed to run in the cloud
  - Machines could be either physical or virtual
- In particular, some manages data of tremendous applications (called ***tenants***)
  - A.k.a. ***multi-tenant DBMS***
- Is MySQL a cloud database?
  - I can run MySQL in a Amazon EC2 VM instance
  - No

What's the Difference?

- Ideally, in addition to all features provided by a traditional database, a cloud database should ensure ***SAE***:
- ***High Scalability***
  - High max. throughput (measured by Tx/Query per second/minute)
  - Horizontal, using commodity machines
- ***High Availability***
  - Stay on all the time, despite of machines/network/datacenter failure
- ***Elasticity***
  - Add/shutdown machines and re-distribute data on-the-fly based on the current workload

Outline

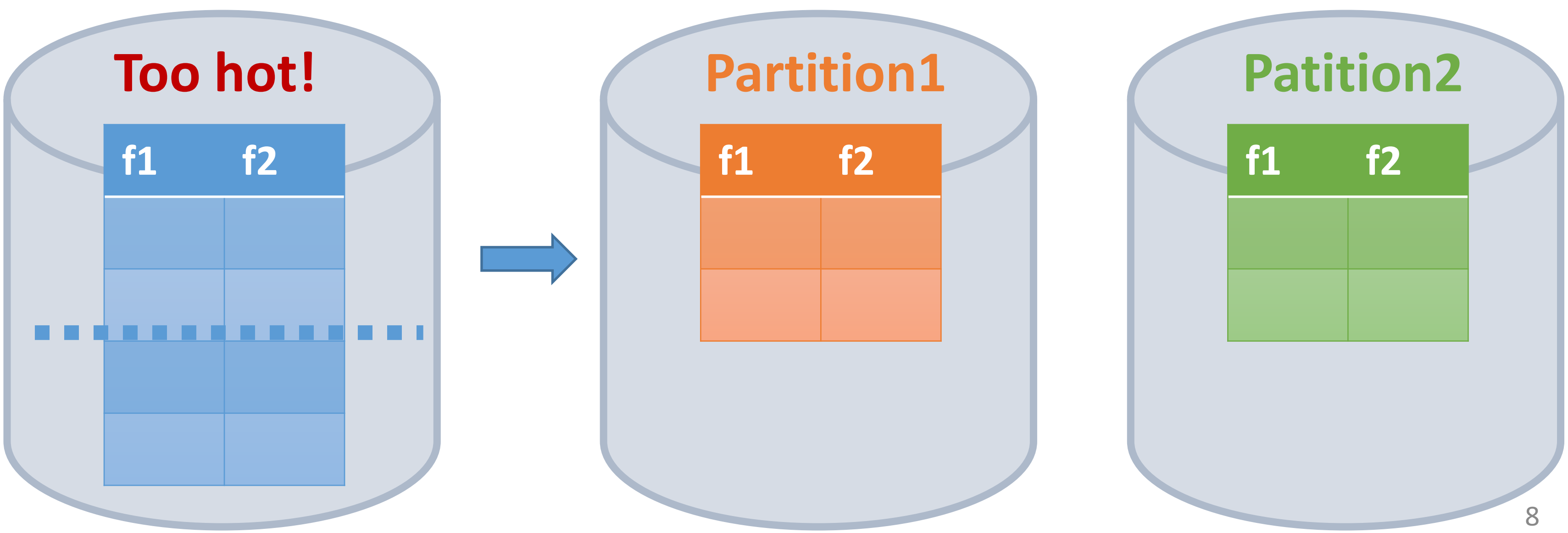
- Goal for Cloud RDBMSs
  - Scalability
  - Availability
  - Elasticity
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

Outline

- Goal for Cloud RDBMSs
  - Scalability
  - Availability
  - Elasticity
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

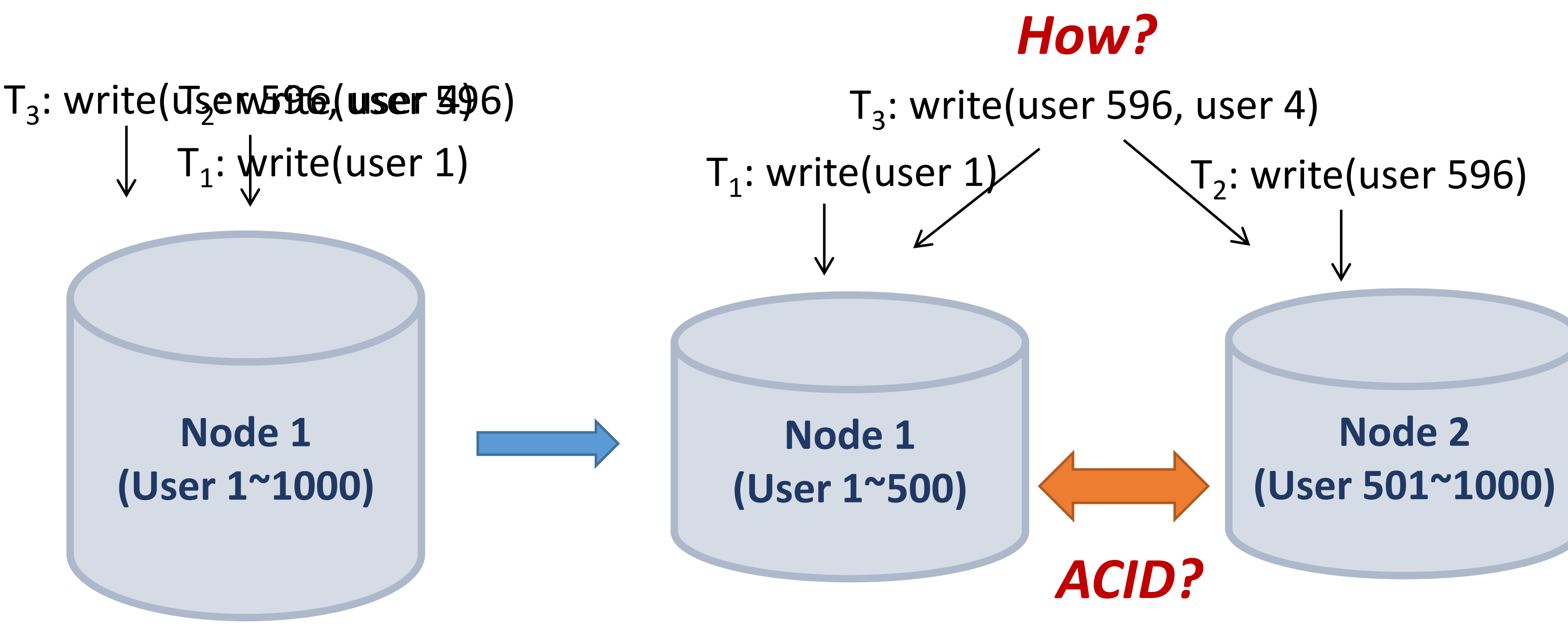
Scalability through Data Partitioning

- ***Partition*** your hot tables
  - Either horizontally or vertically
  - Distribute read/write load to different servers



Complications in Distributed DBs

- Records spread among partitions on different servers



Complications in Distributed DBs

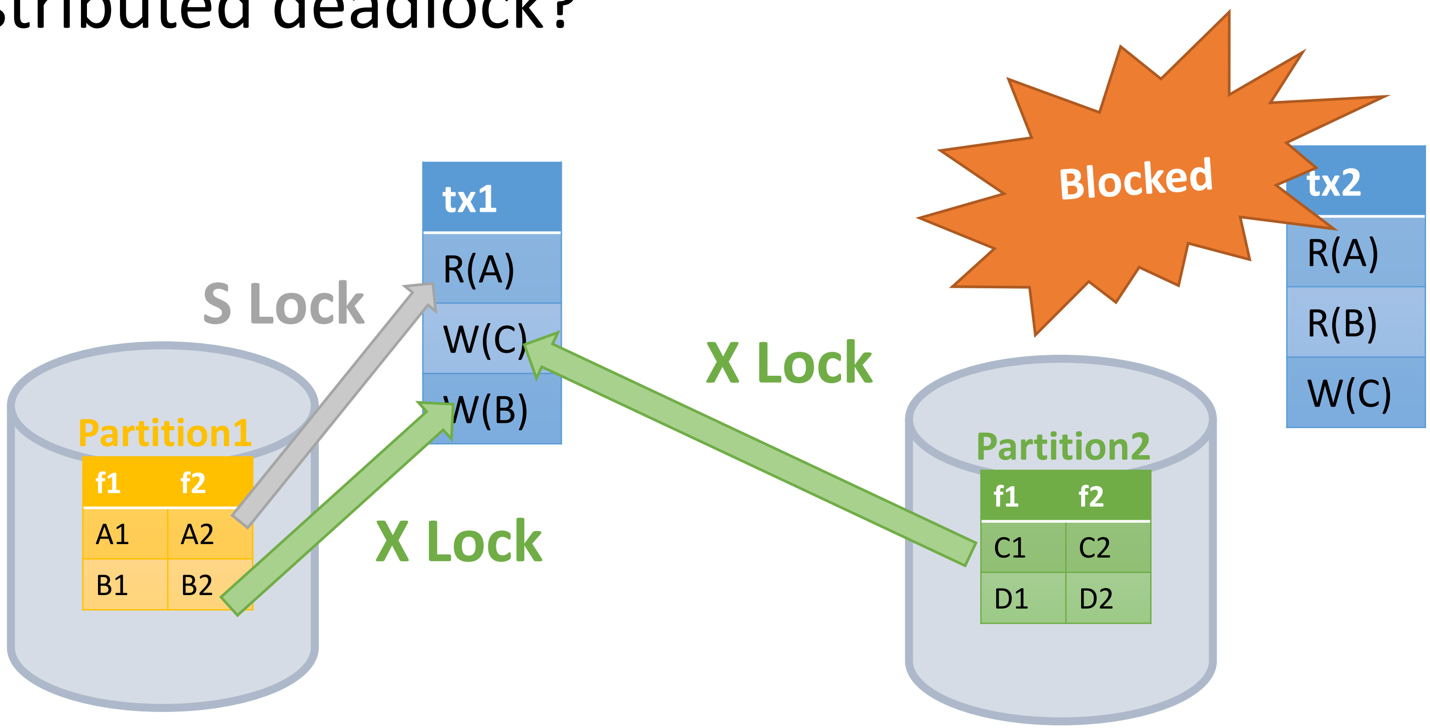
- Records spread among partitions on different servers
- Distributed metadata manager
- Distributed query processor
  - Best global-plan and its local-plans?
- ***Distributed transactions***
  - ACID of a global-transaction T and its local-transactions {Ti}?

Isolation Revisited

- Requires a distributed CC manager
- For 2PL
  - Dedicated lock server, or
  - Primary server for each lock object (***Distributed S2PL***)
- For timestamp and optimistic CC
  - The problem is how to generate the global unique timestamps
  - E.g., “local_counter@server_ID”
  - To prevent one server counts faster, each server increments its own counter upon receiving a timestamp from others

Distributed S2PL

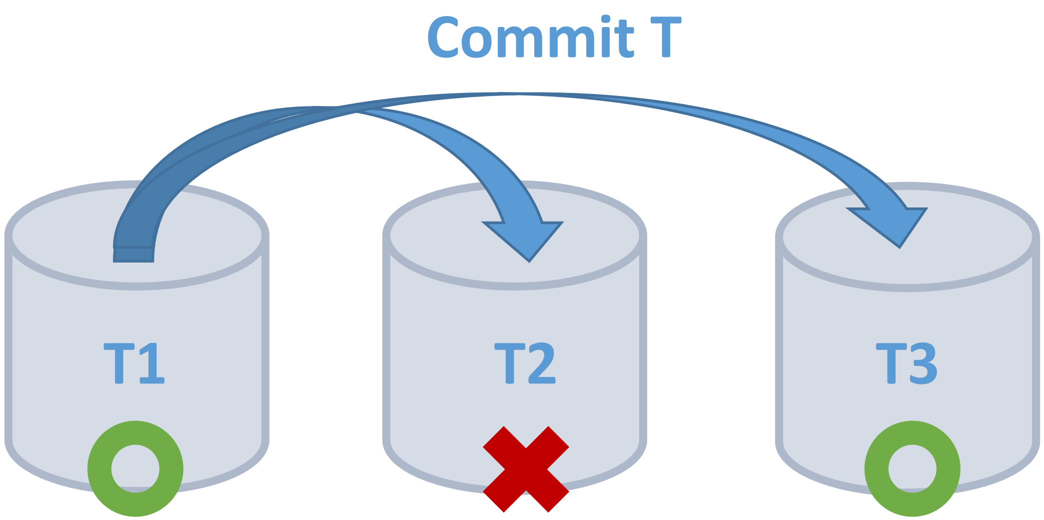
- Primary server of an object: machine owning the corresponding partition
- Distributed deadlock?



Atomicity Revisited

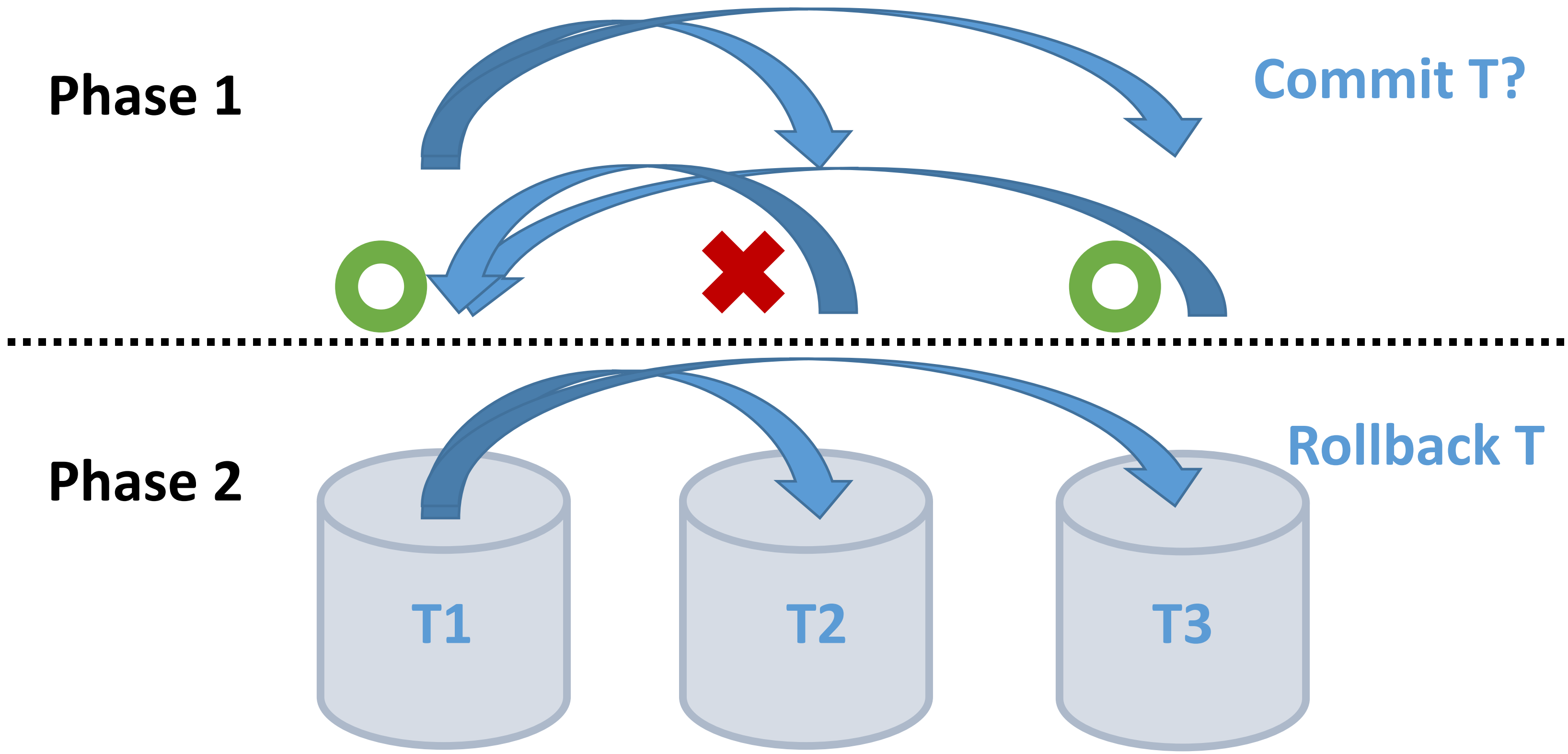
- Committing T means committing all local-transactions
- If any local-transaction rolls back, then T should roll back
  - When will this happen?
  - ACID violation (e.g., in OCC)
  - Deadlock
  - Node failure (detected by some other nodes such as replica)

One-Phase Commit



- If T2 rolls back (due to ACID violation or failure), then T is partially executed, violating atomicity
  - The effect of T1 and T3 cannot be erased due to durability

Two-Phase Commit



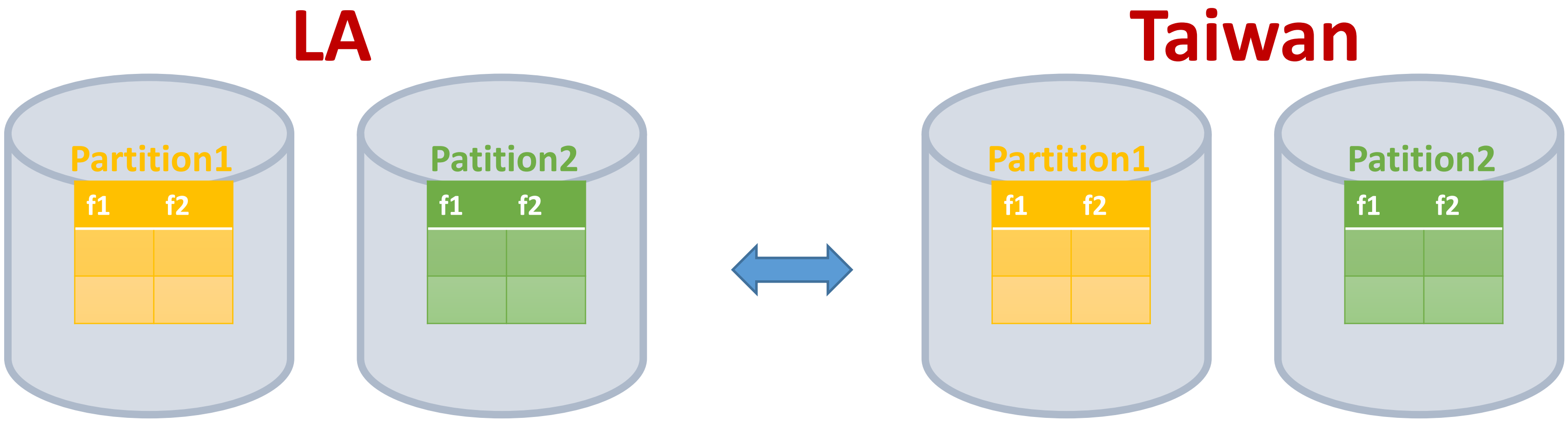
- Drawback: long delay
  - T blocks all conflicting txs and reduces throughput
- Partition helps only when the overhead of communication (2PC) < overhead of slow I/Os

Outline

- Goal for Cloud RDBMSs
  - Scalability
  - Availability
  - Elasticity
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

Availability

- ***Replicate*** all tables across servers
  - If servers in one region fails, we have spare replicas
- Ideally, across geographically-separated regions
  - To deal with disaster

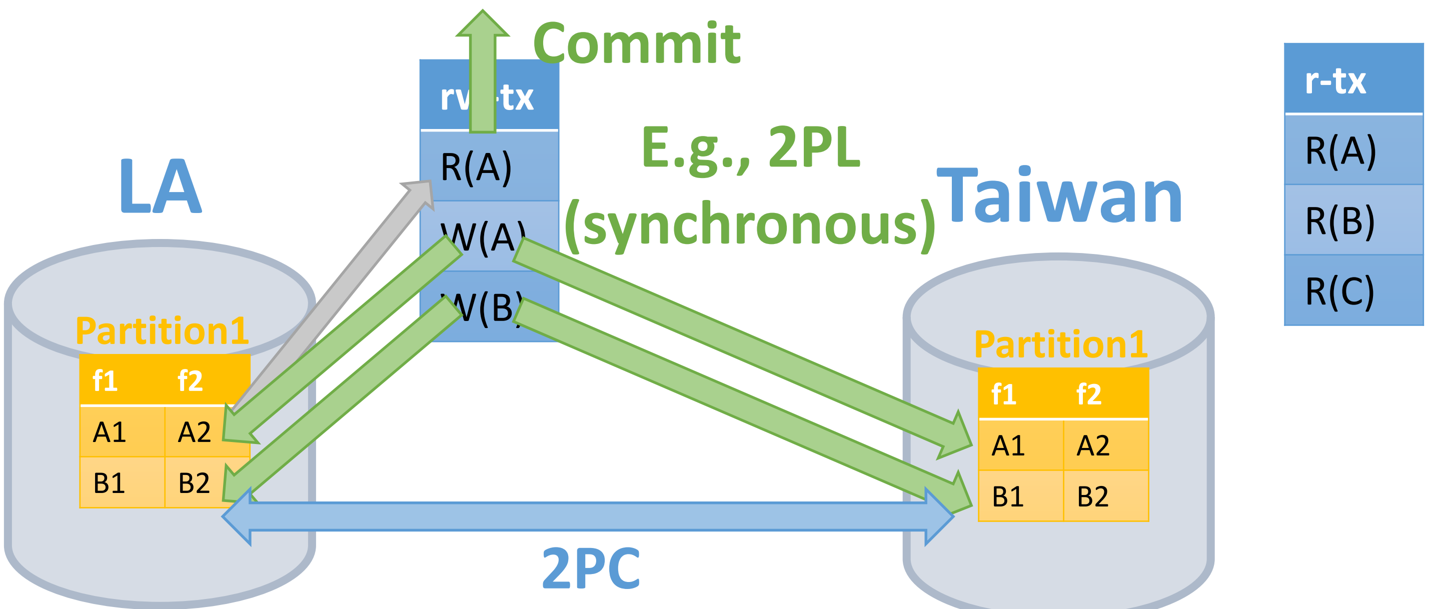


Consistency Revisited

- Consistency
  - Txs do not result in violation of rules you set
- In distributed environments, consistency also means "all replicas ***remain the same*** after executing a tx"
  - Tx reads local, writes all (R1WA)
  - Side-benefit: a read-only tx can be on any replica
- Changes made by a tx on a replica need to be propagated to other replicas
- When? ***Eager vs. Lazy***
- By whom? ***Master/Slave vs. Multi-Master***

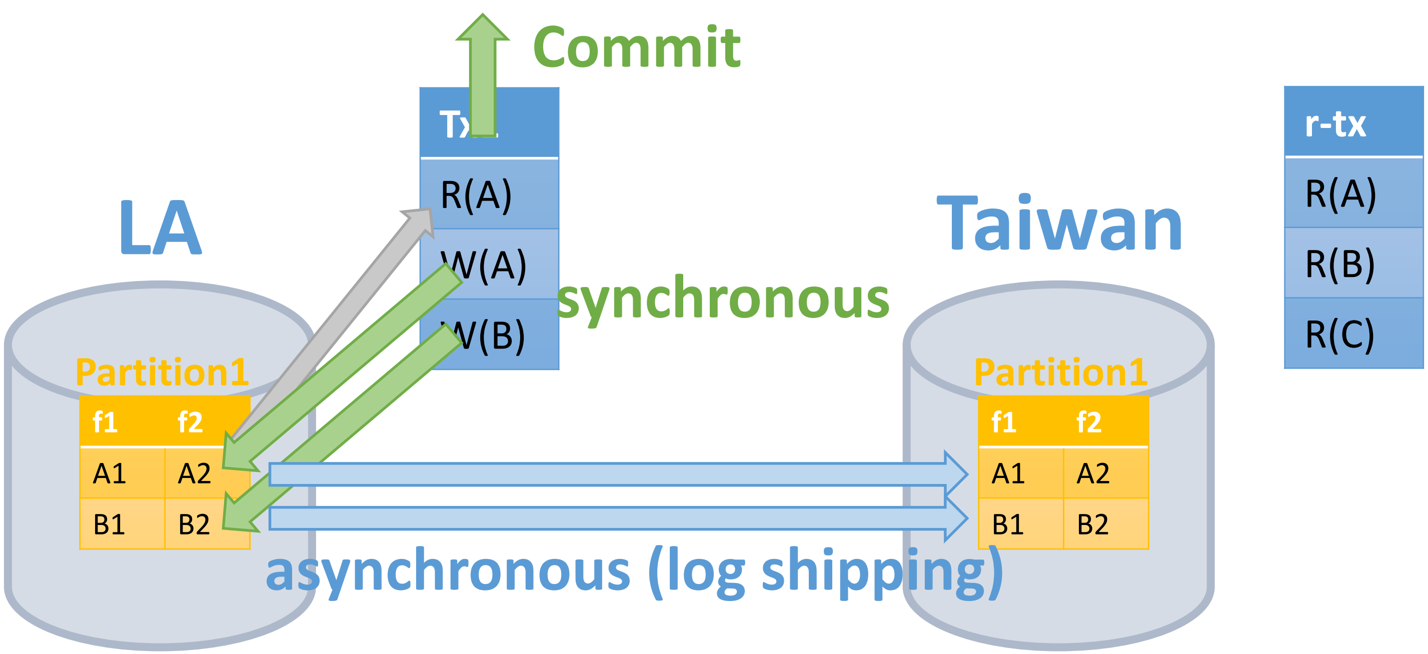
Eager Replication

- Each write operation must completes on all replicas ***before a tx commits***
  - 2PC required
  - Failure on any replica causes tx's rollback
- Slow tx, but strong consistency



Lazy Replication

- Writes complete locally, but are propagated to remote replicas ***in the background*** (with a lag)
  - Usually by shipping a batch of logs
  - Fast tx, but eventual consistency



Who Writes?

- ***Master/Slave*** replication
  - Writes of a record are routed to a specific (called master) replica
  - Reads to others (slave replicas)
- ***Multi-Master*** replication
  - Writes of a record can be routed to any replica
  - I.e., two writes of the same records may be handled by ***different*** replicas

The Score Sheet

	Eager MM	Lazy M/S	Lazy MM
Consistency	Strong	***Eventual***	***Weak***
Latency	***High***	Low	Low
Throughput	***Low***	High	High
Availability upon failure	Read/write	***Read-only***	Read/write
Data loss upon failure	None	***Some***	***Some***
Reconciliation	No need	No need	***User or rules***
Bottleneck, SPF	None	***Master***	None

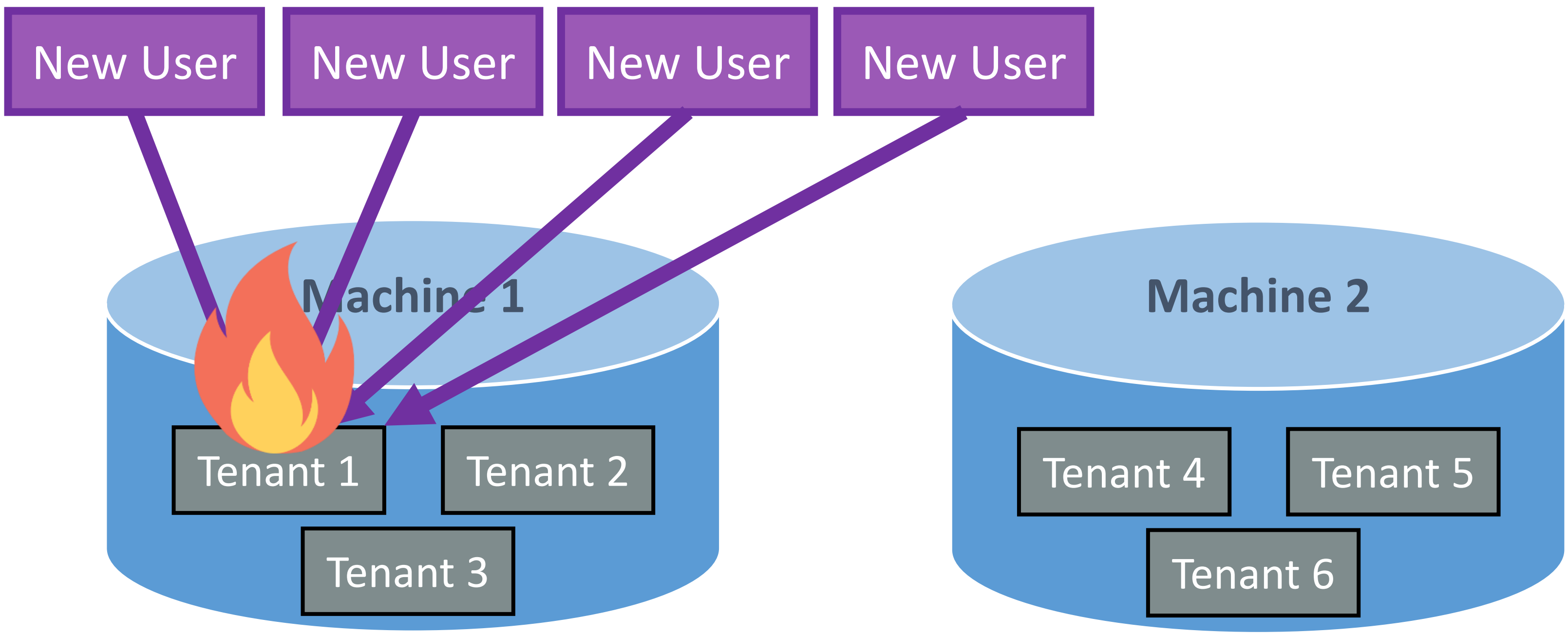
- Eager M/S and MM make no much difference with 2PL + 2PC
- Lazy MM needs reconciliation of conflicting writes
  - Either by user or rules, e.g., last-write-wins

Outline

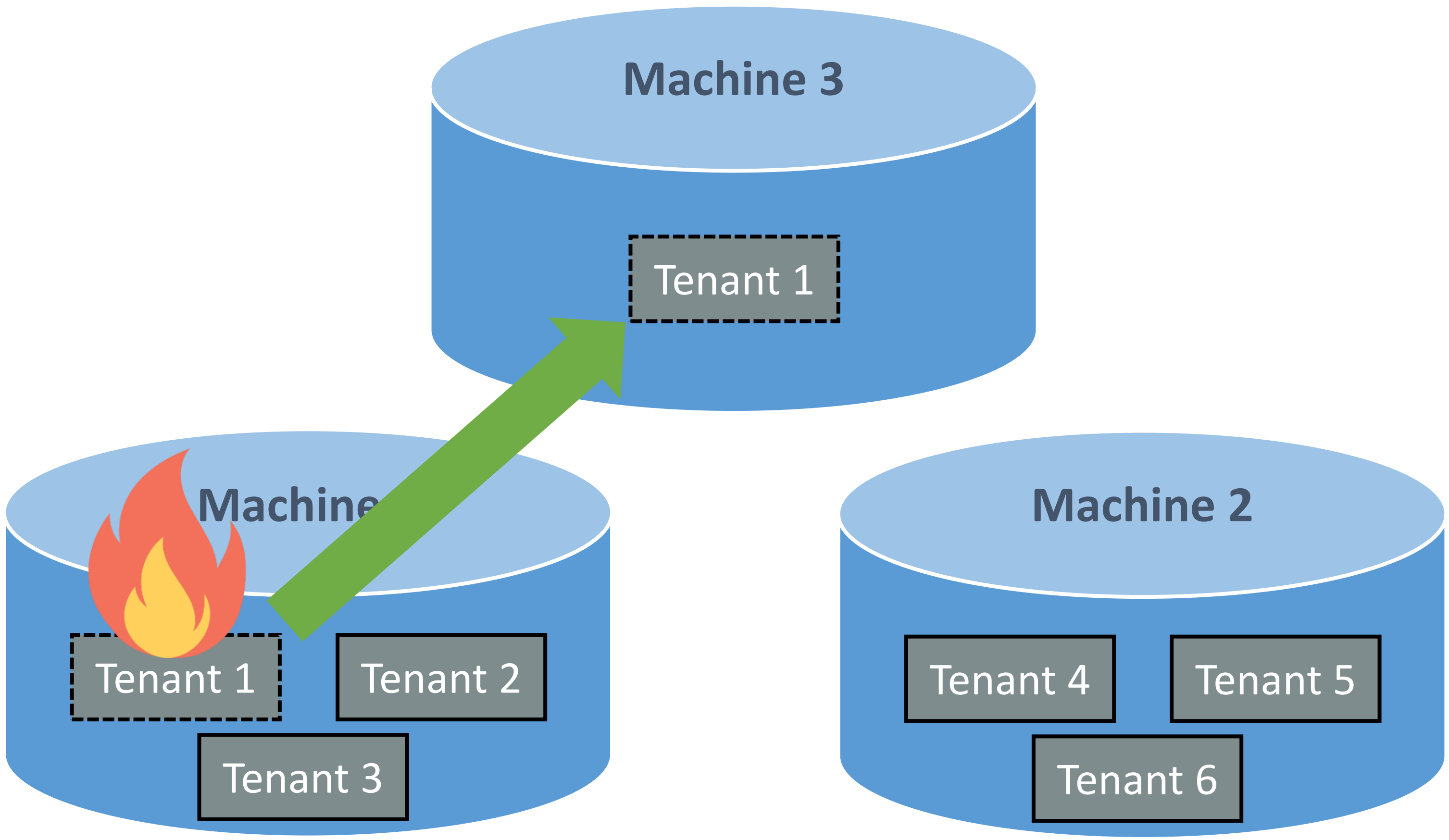
- Goal for Cloud RDBMSs
  - Scalability
  - Availability
  - Elasticity
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

Motivation: Hot Tenants

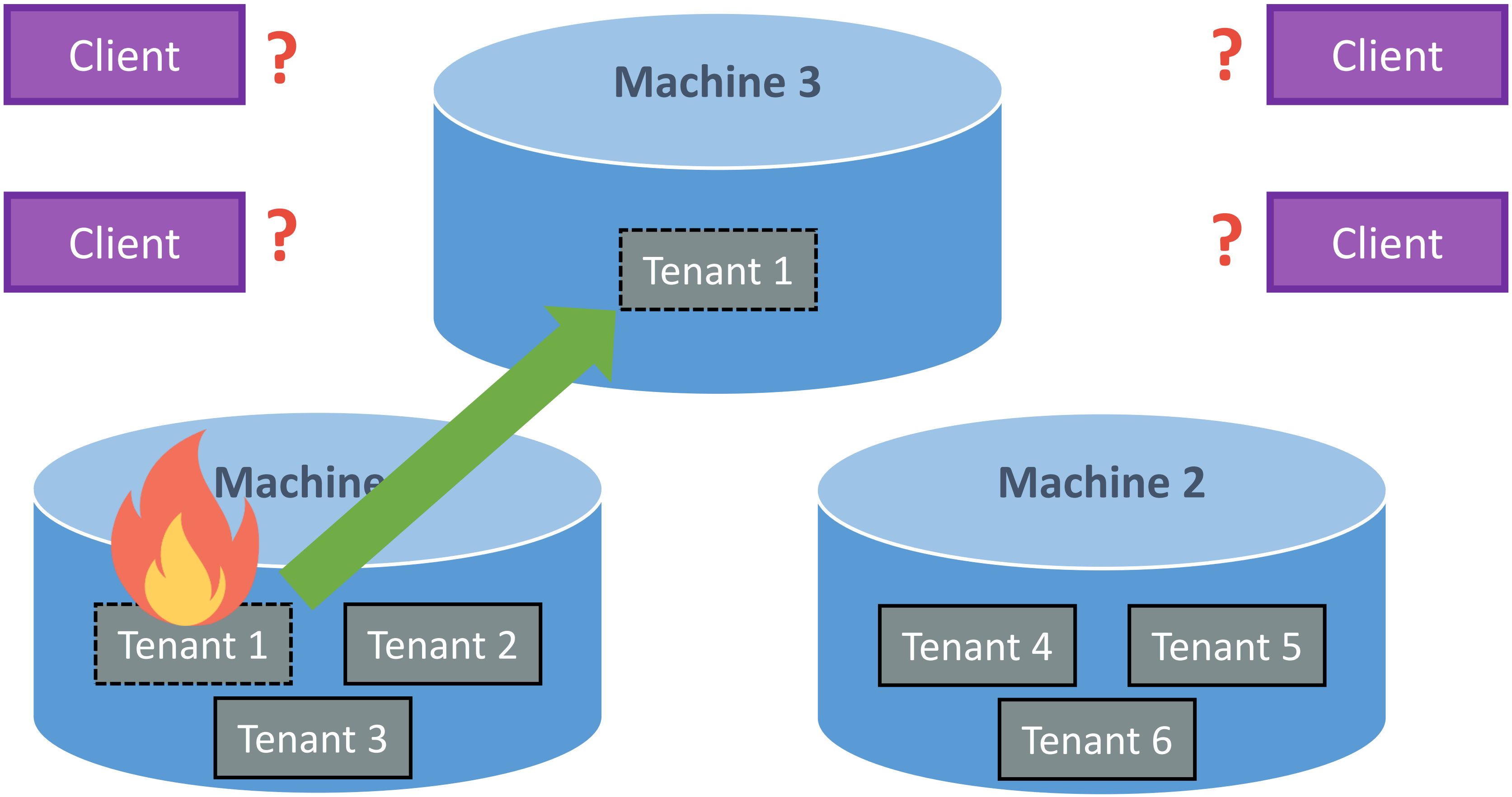
- An application gains flash crowds originating from viral popularity.



Solution: Adding More Resource by Increasing Provisioning Machines



How to Move Data On The Fly While Keeping Serving Transactions?



Two Folds of This Problem

- How to re-partition data?
  - Which records to be chosen?
  - Where to put those records?
- How to migrate data?
  - How to move data such that no transaction is interrupted?

- These are active research topics

Current Research Status

- Determinism solves almost all the problems!
  - [SIGMOD'12] Calvin
  - [SIGMOD'16] T-Part
  - [VLDB'19] MgCrab
  - [SIGMOD'21?] Hermes

- We call a cloud DBMS that has high scalability as a NewSQL DBMS.

Current NewSQL Systems

		Year Released	Main Memory Storage	Partitioning	Concurrency Control	Replication	Summary
New Architectures	**Clustrix** [6]	2006	No	Yes	MVCC+2PL	Strong+Passive	MySQL-compatible DBMS that supports shared-nothing, distributed execution.
	CockroachDB [7]	2014	No	Yes	MVCC	Strong+Passive	Built on top of distributed key/value store. Uses software hybrid clocks for WAN replication.
	Google Spanner [24]	2012	No	Yes	MVCC+2PL	Strong+Passive	WAN-replicated, shared-nothing DBMS that uses special hardware for timestamp generation.
	H-Store [8]	2007	Yes	Yes	TO	Strong+Active	Single-threaded execution engines per partition. Optimized for stored procedures.
	HyPer [9]	2010	Yes	Yes	MVCC	Strong+Passive	HTAP DBMS that uses query compilation and memory efficient indexes.
	MemSQL [11]	2012	Yes	Yes	MVCC	Strong+Passive	Distributed, shared-nothing DBMS using compiled queries. Supports MySQL wire protocol.
	NuoDB [14]	2013	Yes	Yes	MVCC	Strong+Passive	Split architecture with multiple in-memory executor nodes and a single shared storage node.
	SAP HANA [55]	2010	Yes	Yes	MVCC	Strong+Passive	Hybrid storage (rows + cols). Amalgamation of previous TREX, P*TIME, and MaxDB systems.
	VoltDB [17]	2008	Yes	Yes	TO	Strong+Active	Single-threaded execution engines per partition. Supports streaming operators.
Middleware	**AgilData** [1]	2007	No	Yes	MVCC+2PL	Strong+Passive	Shared-nothing database sharding over single-node MySQL instances.
	MariaDB MaxScale [10]	2015	No	Yes	MVCC+2PL	Strong+Passive	Query router that supports custom SQL rewriting. Relies on MySQL Cluster for coordination.
	ScaleArc [15]	2009	No	Yes	Mixed	Strong+Passive	Rule-based query router for MySQL, SQL Server, and Oracle.
DBaaS	**Amazon Aurora** [3]	2014	No	No	MVCC	Strong+Passive	Custom log-structured MySQL engine for RDS.
	ClearDB [5]	2010	No	No	MVCC+2PL	Strong+Active	Centralized router that mirrors a single-node MySQL instance in multiple data centers.

Andrew Pavlo and Matthew Aslett. 2016. What's Really New with NewSQL? *SIGMOD Rec.* 45, 2 (June 2016), 45–55.

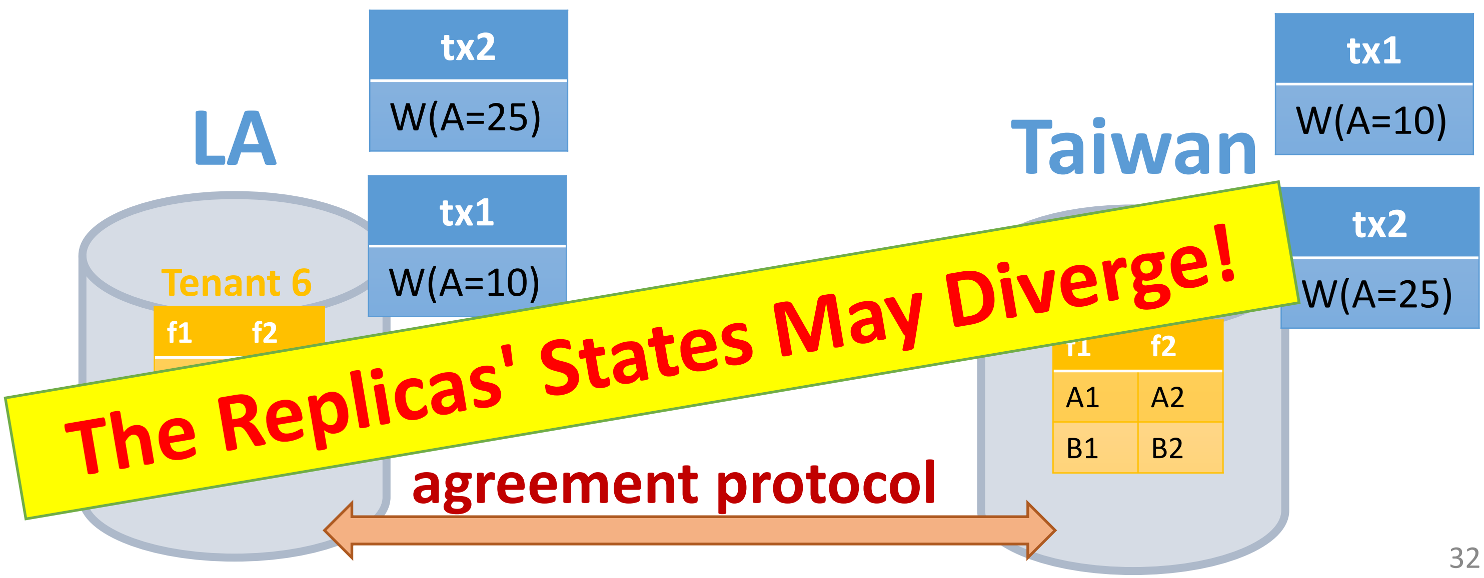
Outline

- Goal for Cloud RDBMSs
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

What is a deterministic DDBMS?

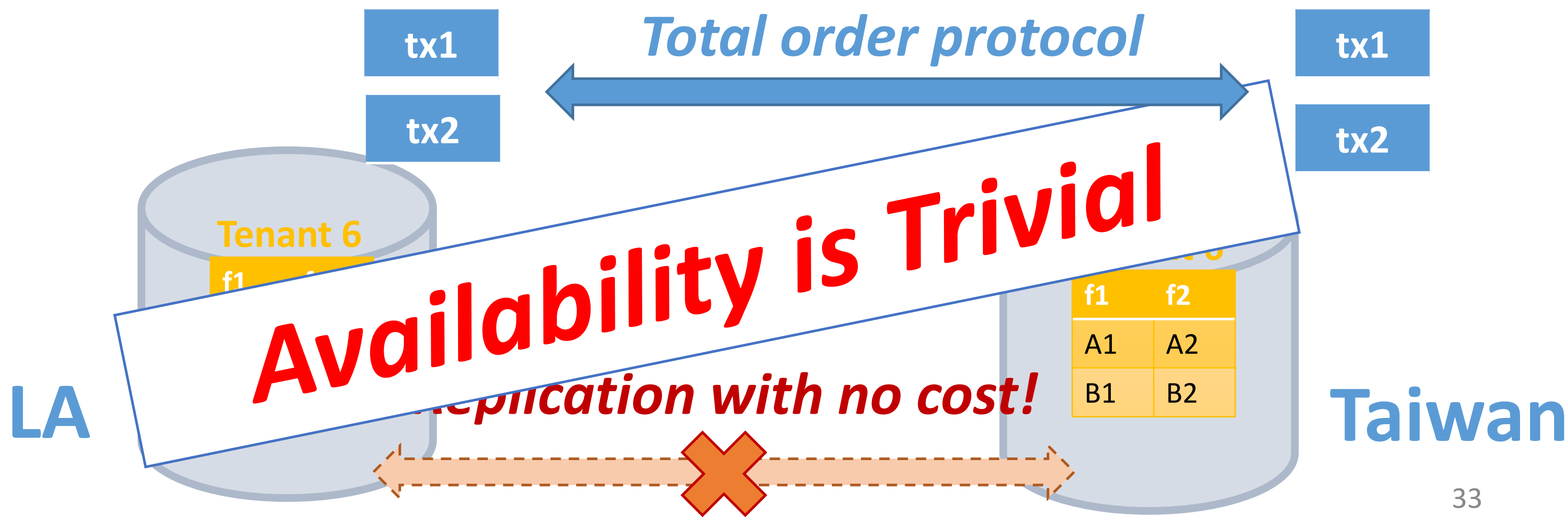
What Is A Non-Deterministic DDBMS?

- Given Tx_1 and Tx_2, traditional DBMS guarantees ***some*** serial execution: Tx_1 -> Tx_2, or Tx_2 -> Tx_1
- Given a collection of requests/txs, DDBMS leaves the data (outcome) non-deterministically due to
  - Delayed requests, CC (lock granting), deadlock handling, buffer pinning, threading, etc.

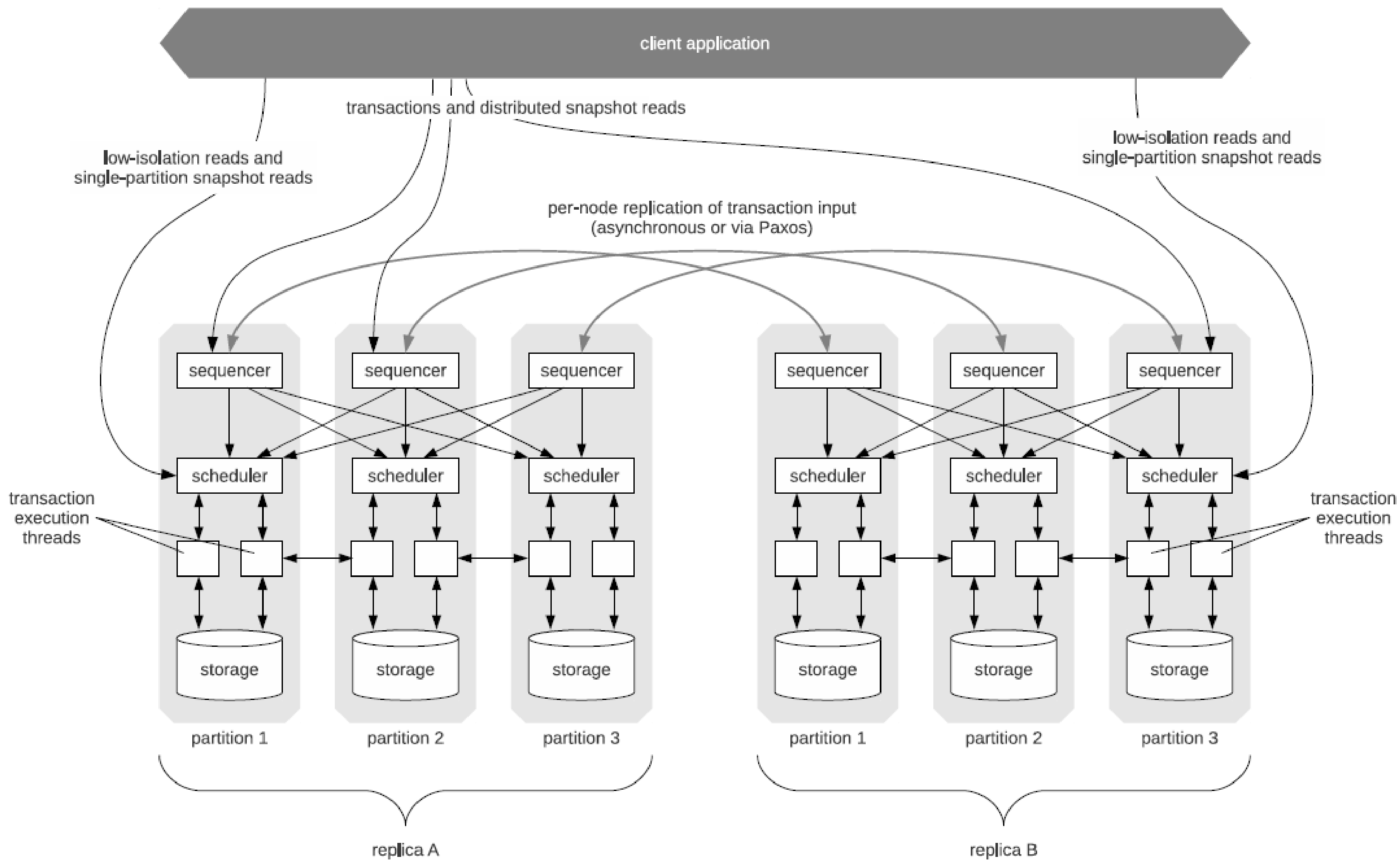


Deterministic DDBMS

- Given a collection of ***ordered*** requests/txs, if we can ensure that
  1) conflicting txs are executed on each replica following that order
  2) each tx is always run to completion (commit or abort by tx logic), even failure happens
- Then data in all replicas will be in same state, i.e., ***determinism***



Architecture of Calvin [SIGMOD'12]

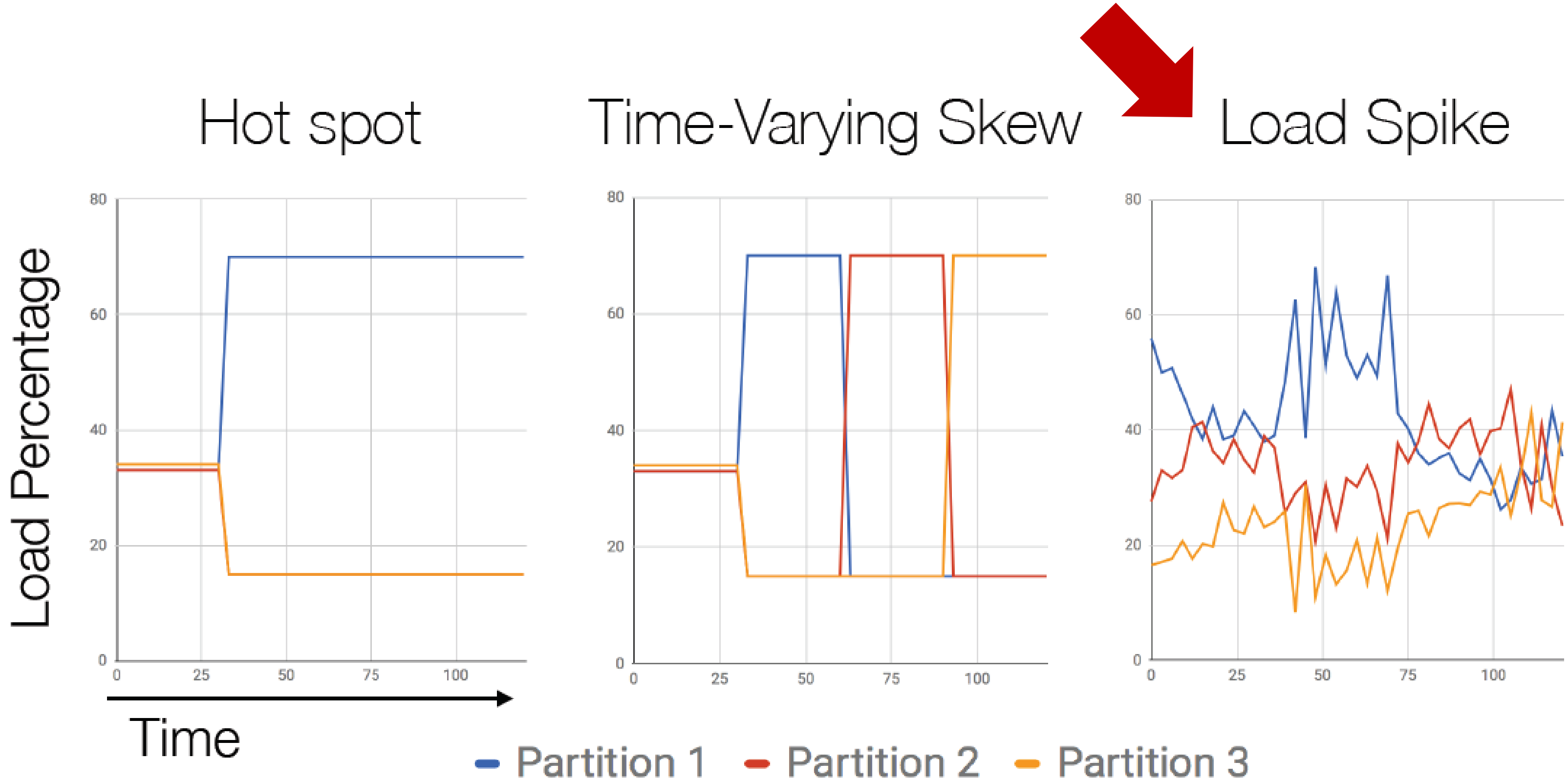


Outline

- Goal for Cloud RDBMSs
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

Motivation – Changing Workloads

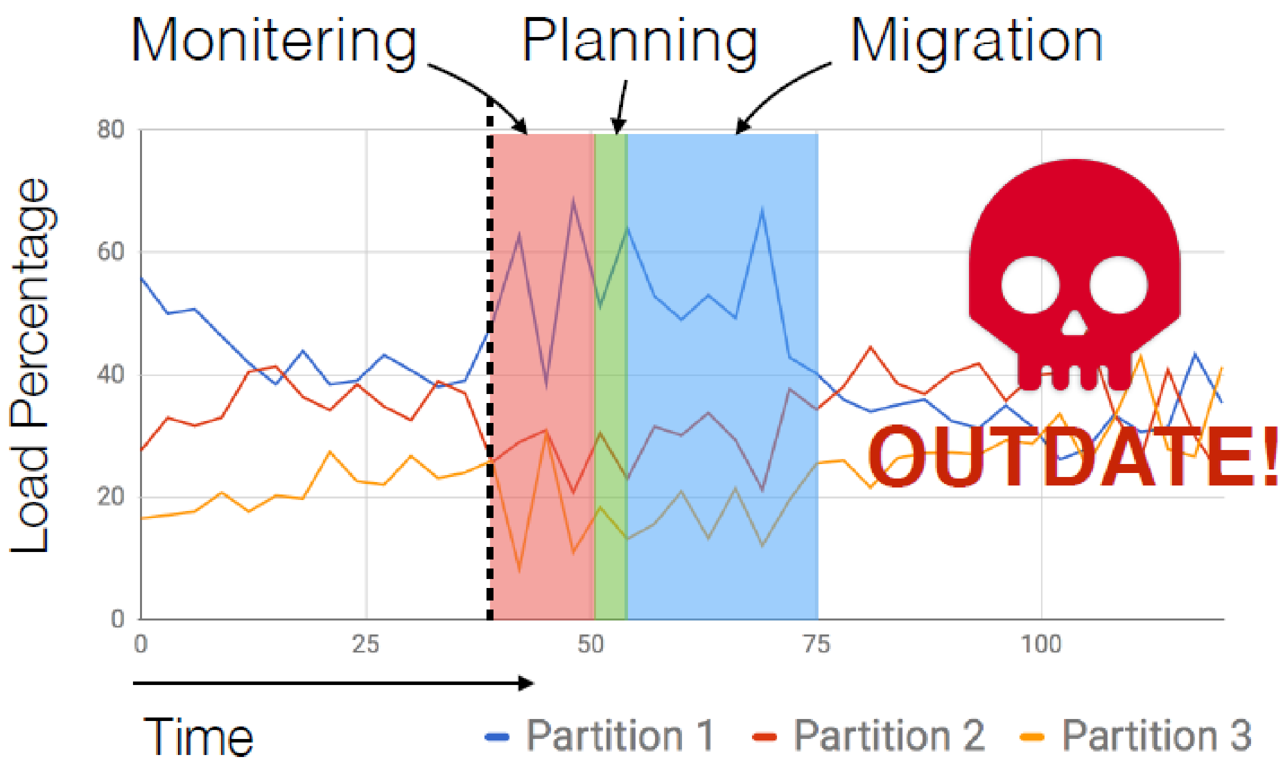
- Workload changes & **Unpredictable** workloads



How to Re-partitioning Data for such workloads?

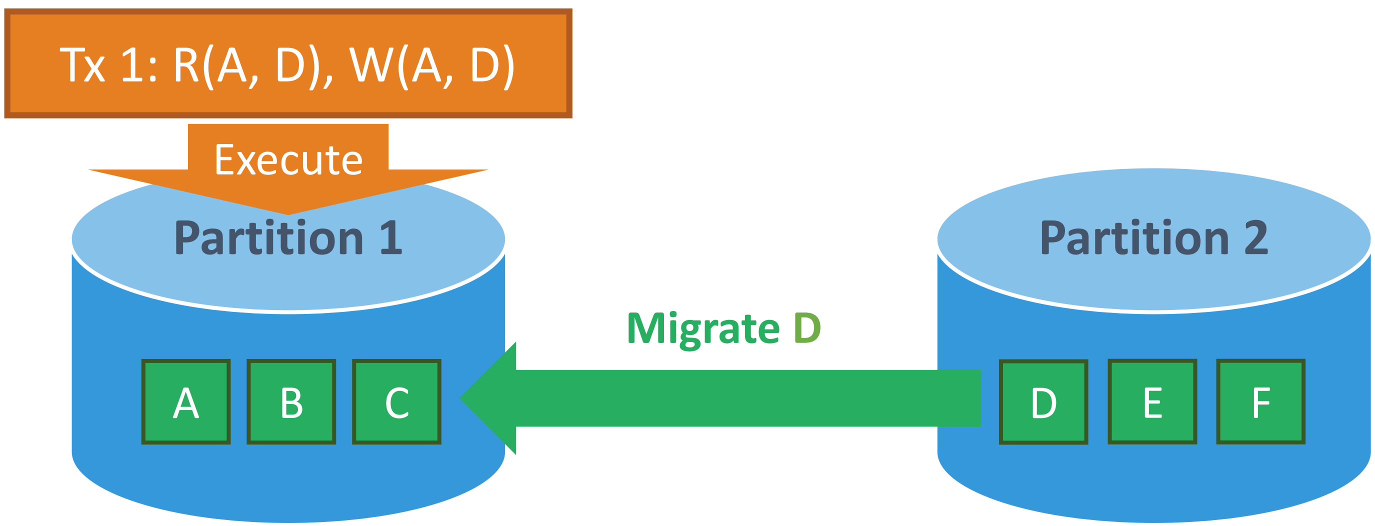
Look-back Approaches

- [VLDB'16] Clay

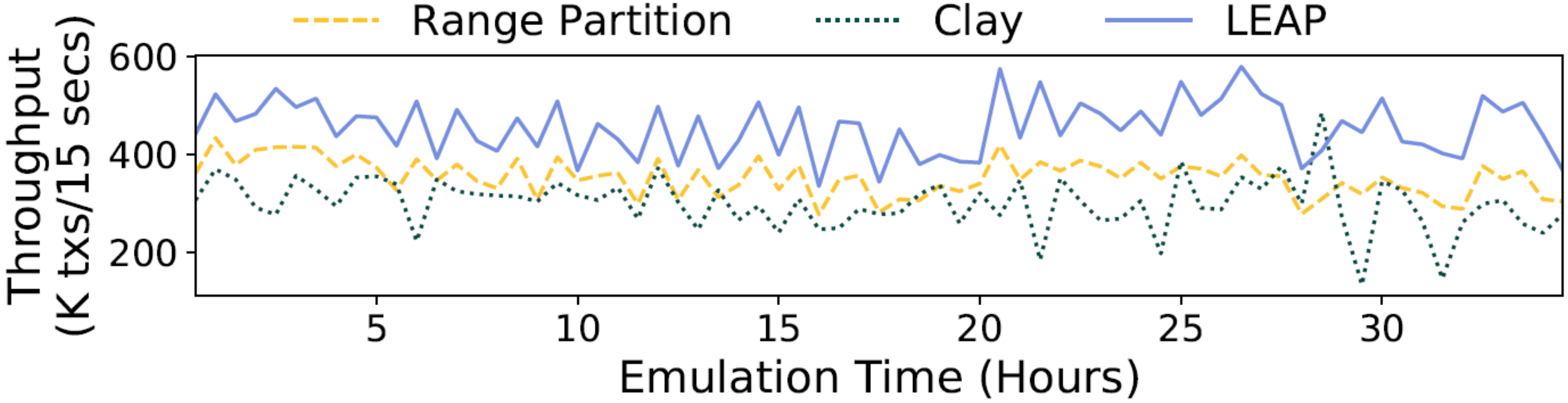


Look-present Approaches

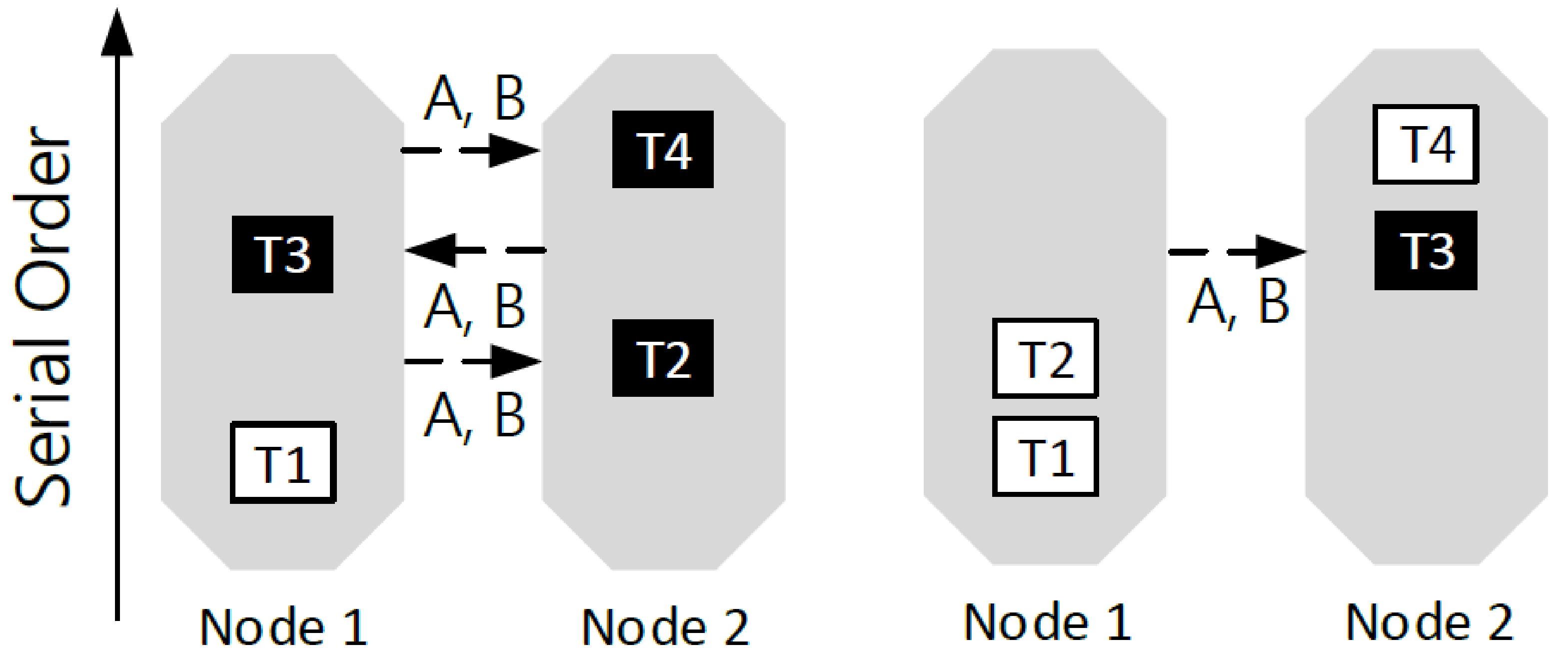
- [SIGMOD'16] LEAP
  - Migrate data to where they are used



Not Good Enough



Ping-pong Problems

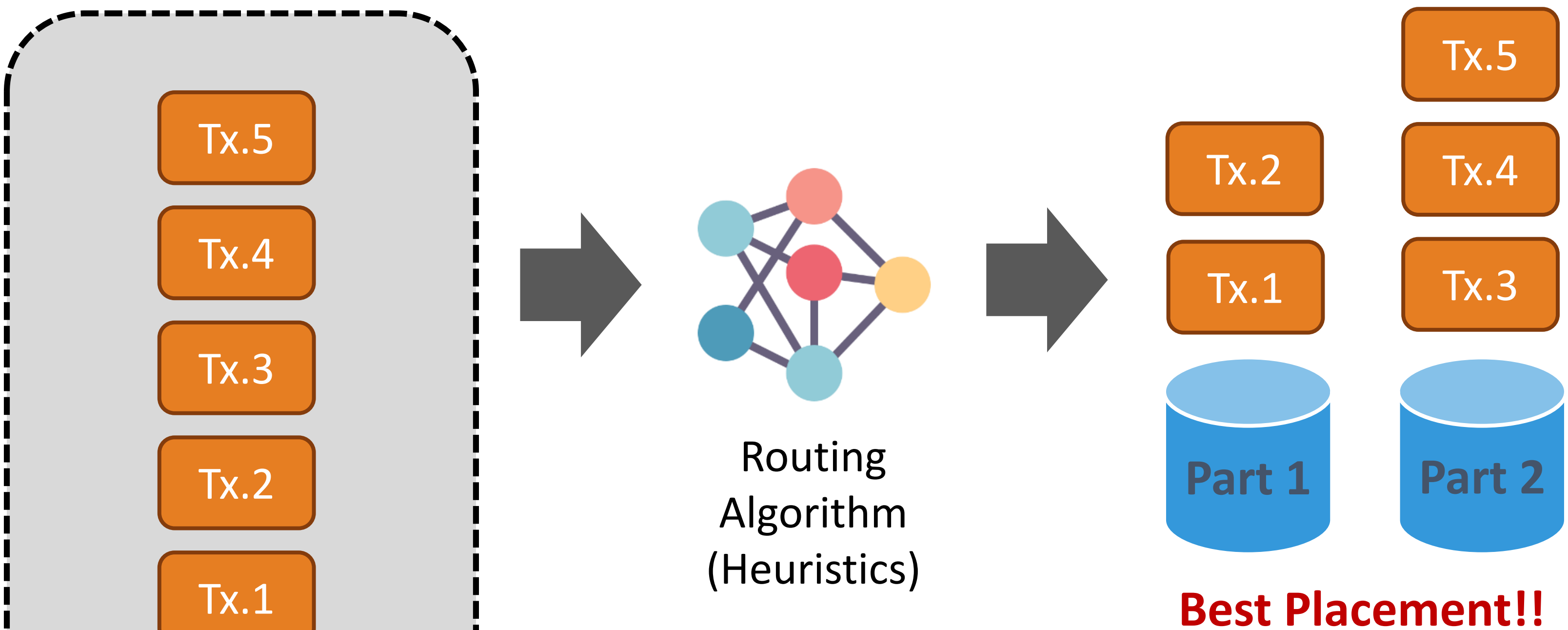


(a) Schedule 1 (b) Schedule 2

How About
Looking into the Future?

Looking into the Future (Batch)

- The prescient routing (heuristics)



=> Minimizes data migrations
=> Balances loads among machines

Outline

- Goal for Cloud RDBMSs
- Deterministic DBMSs - Calvin
- Our Recent Research - Hermes
- Our Next Step: RL-driven On-line Repartitioning

Problem of the Routing

- A heuristics way
  - No guarantee that performance will be improved
- With a restriction
  - We fixed the order of transactions to shrink the search space
- Search time: $O(a^2 b^2 n)$
  - a: max # of reads per tx
  - b: batch size
  - n: # of machine nodes

- Better algorithm?

Reinforcement Learning!!

- Advantages
  - Learning by trying => ability to adapt workloads
  - Exploring possibility
  - GPU-accelerated (deep reinforcement learning)
- How to model?
  - A working topic
  - Action? reward? states?

Assigned Readings

- Deterministic DBMS
  - Thomson, Alexander, and Daniel J. Abadi. "The case for determinism in database systems." Proceedings of the VLDB Endowment 3.1-2 (2010): 70-80.
  - Thomson, Alexander, et al. "Calvin: fast distributed transactions for partitioned database systems." Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. 2012.
- Deep Reinforcement Learning
  - Prof. Wu's DL Lecture 16~17
- Our Research
  - Hermes (I will give you a copy)