Replication-driven Live Reconfiguration for Fast Distributed Transaction Processing

Xingda Wei, Sijie Shen,
Rong Chen, Haibo Chen

Institute of Parallel and Distributed Systems
Shanghai Jiao Tong University, China

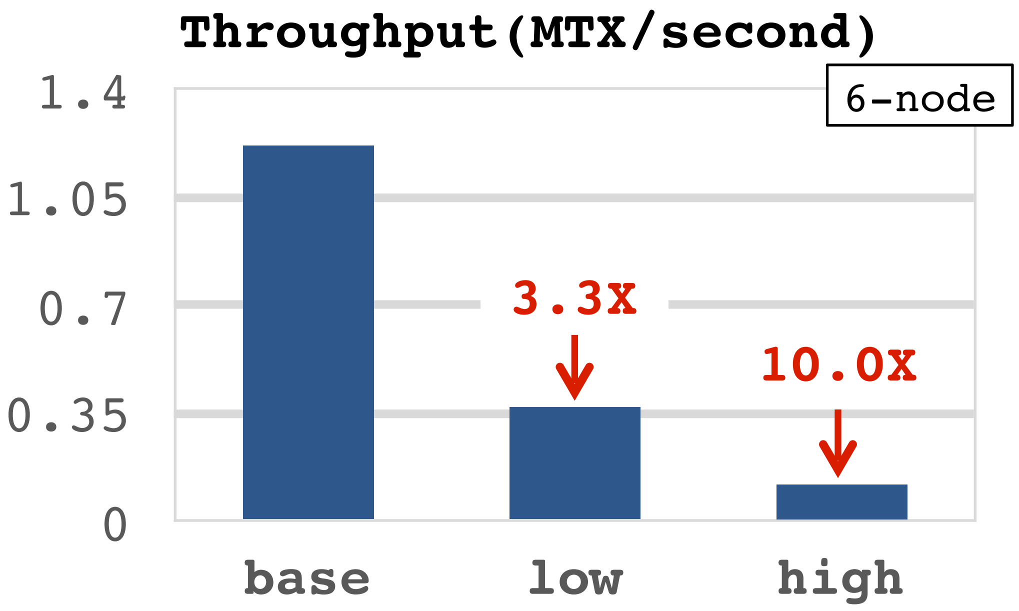
Transactions: Key pillar for many systems

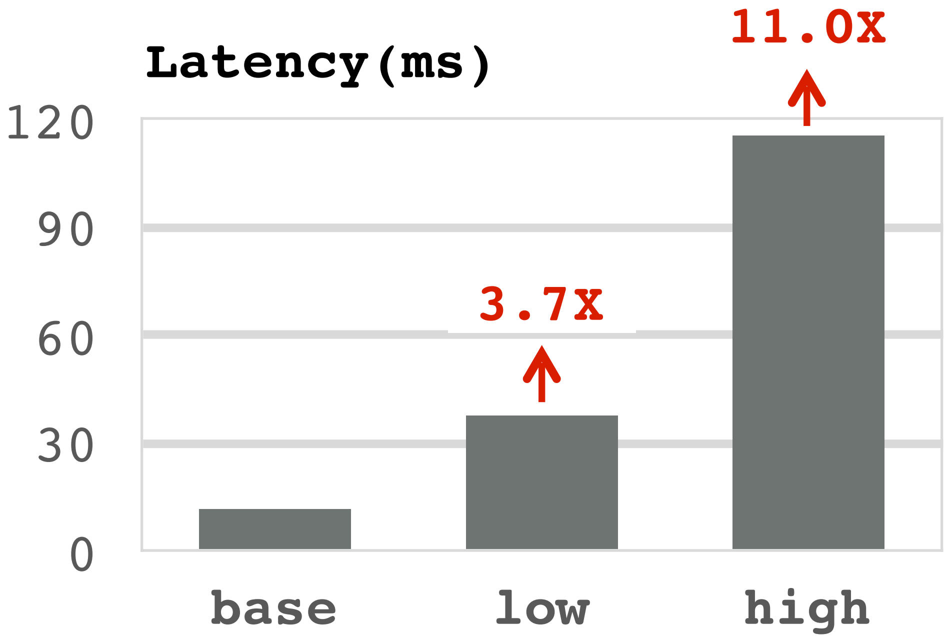
$9.3 billion/day

11. 6 million payments/day

Skewed workloads hurt performance

- Modern in-memory transactions are **fast**
- TXs can **scale-out** on **balanced** workloads
  - But can fail with skewed workloads

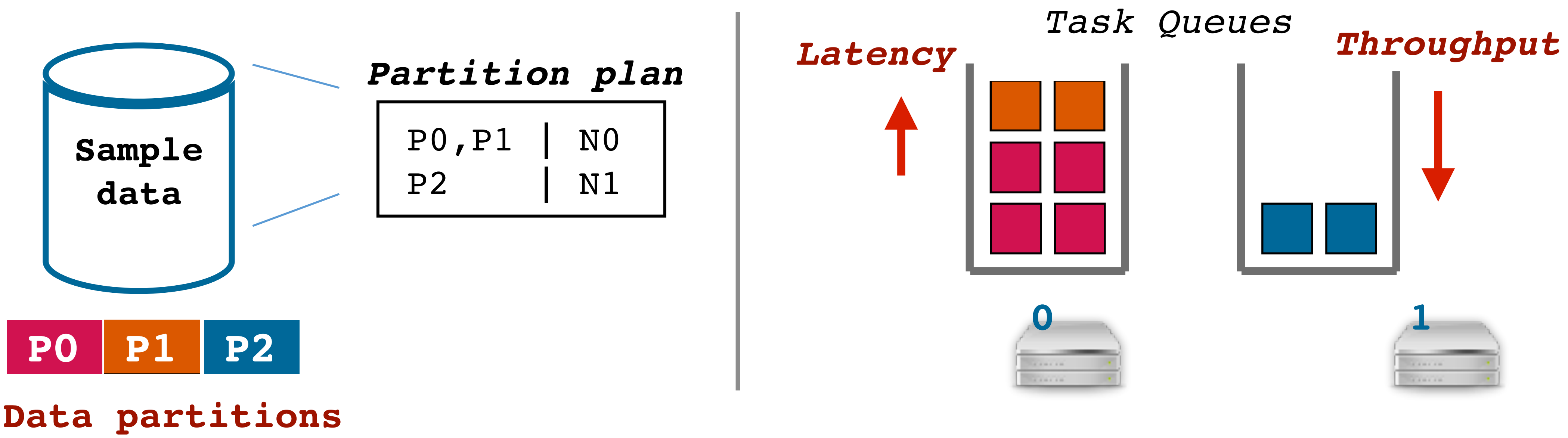




Skewed **TPCC** workloads on **DrTM+R**[Eurosys'16]

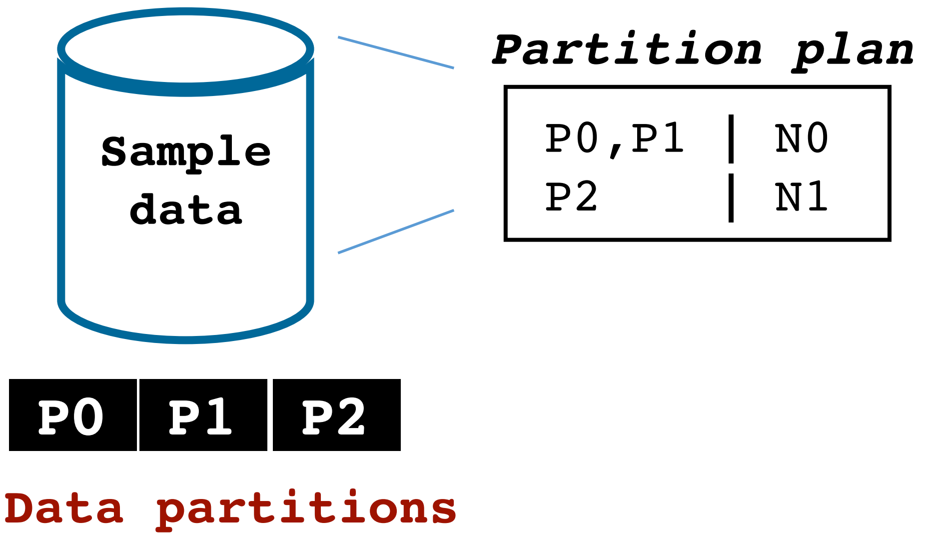
Skewed workloads hurt performance

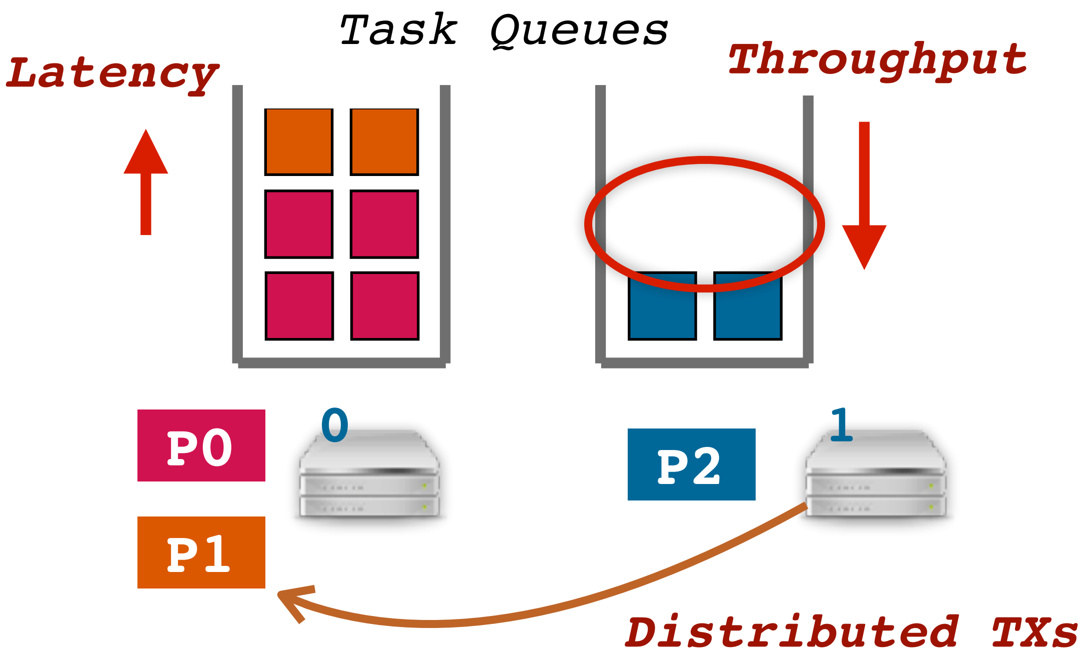
- Unbalanced workload: Idle worker



Skewed workloads hurt performance

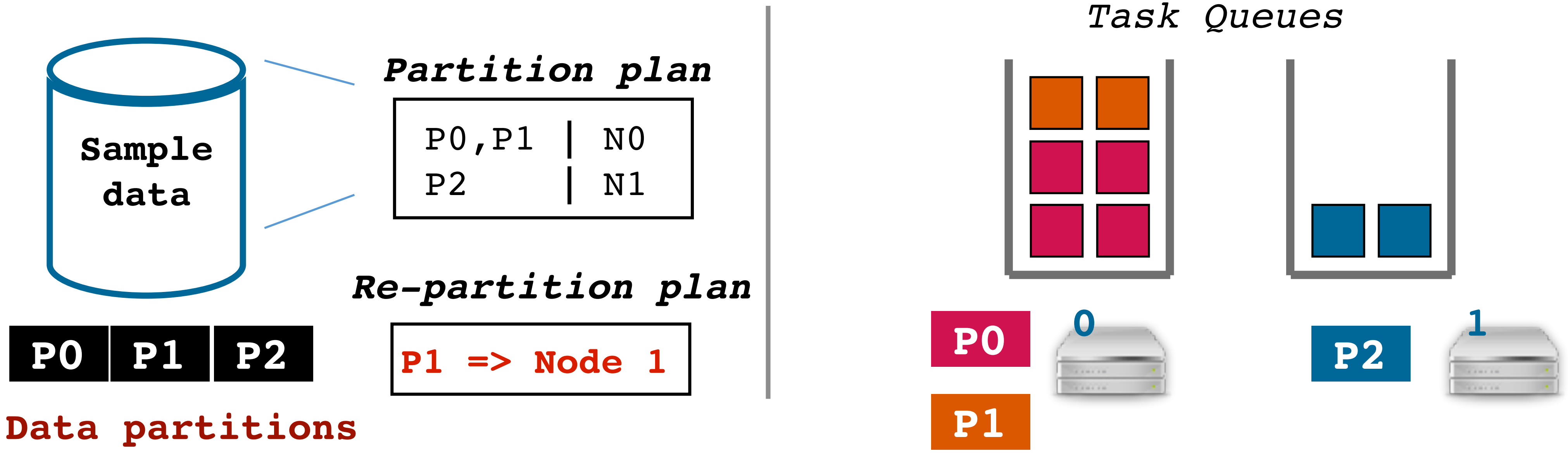
- Unbalanced workload: Idle worker
- More distributed TX ratio, aborts





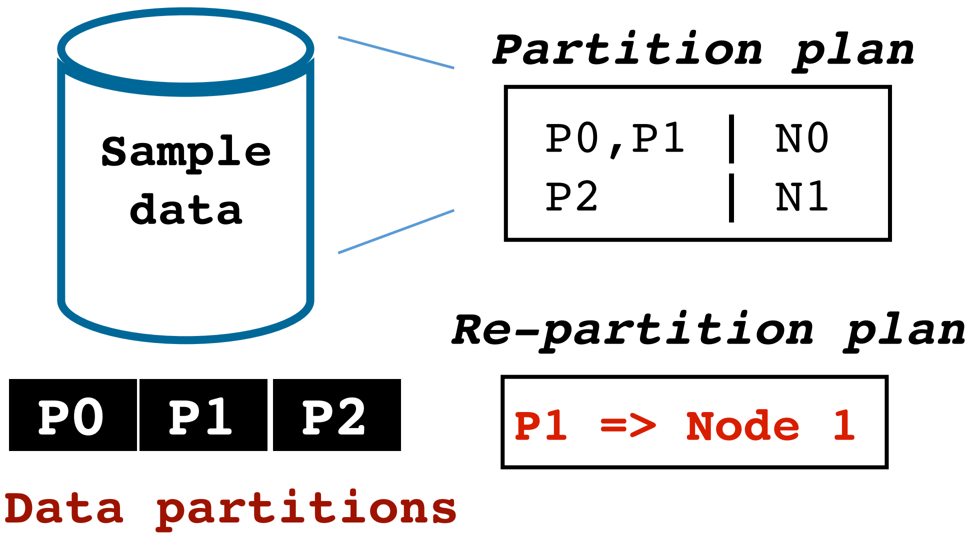
Solution: Live Reconfiguration

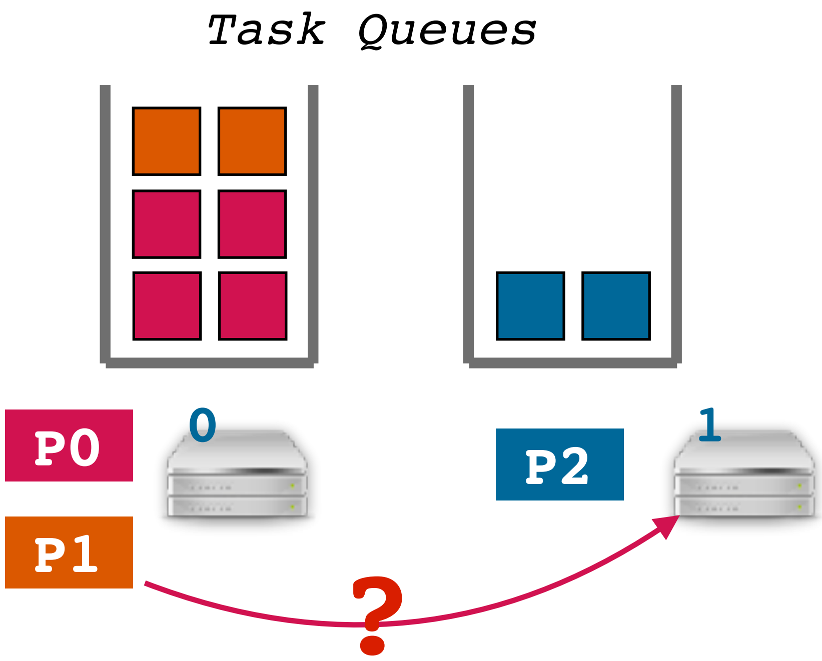
- E-store[VLDB'14] **repartitions** the database to **balance** the workload on each server



Efficient live reconfiguration

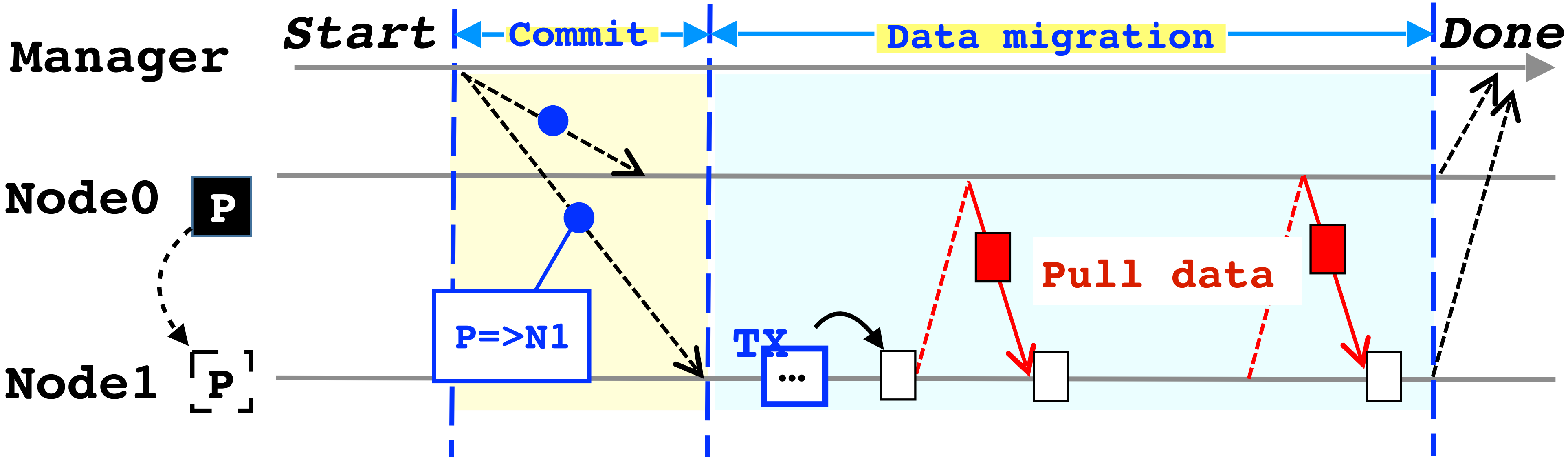
- Generate the re-partition plan **is fast**
- How to **lively** migrate the data?





SOL#1 Migrating data with Post-copy

- E-store uses **Squall**[SIGMOD'15] to migrate data

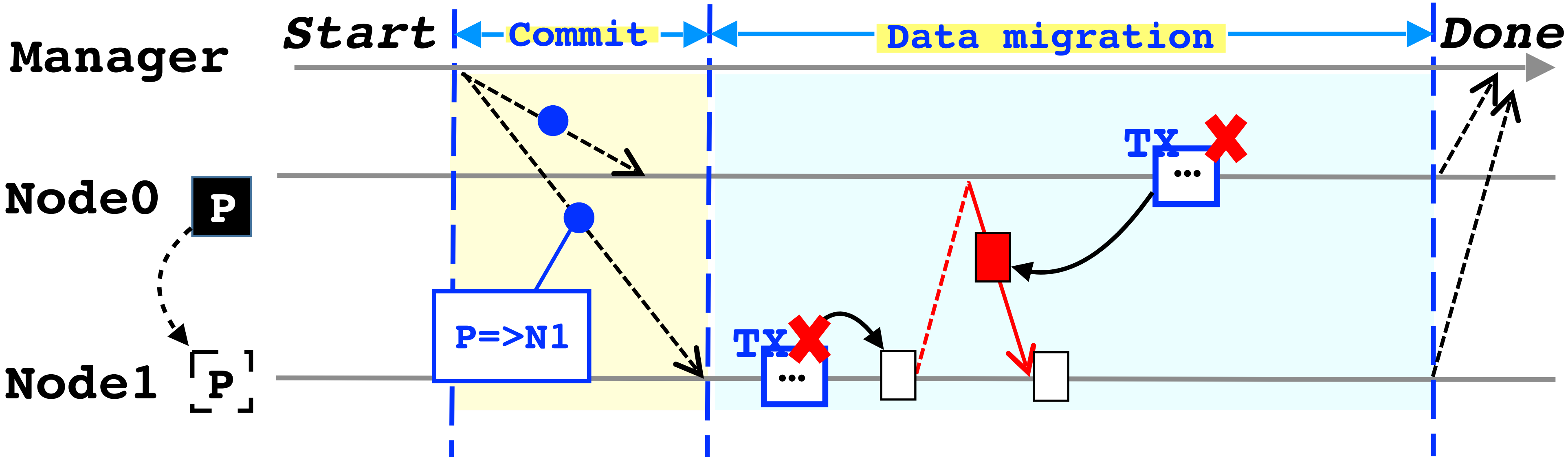


① Commit the plan (i.e P is at N1)

② Pull data on-demand **or** asynchronously

SOL#1 Migrating data with Post-copy

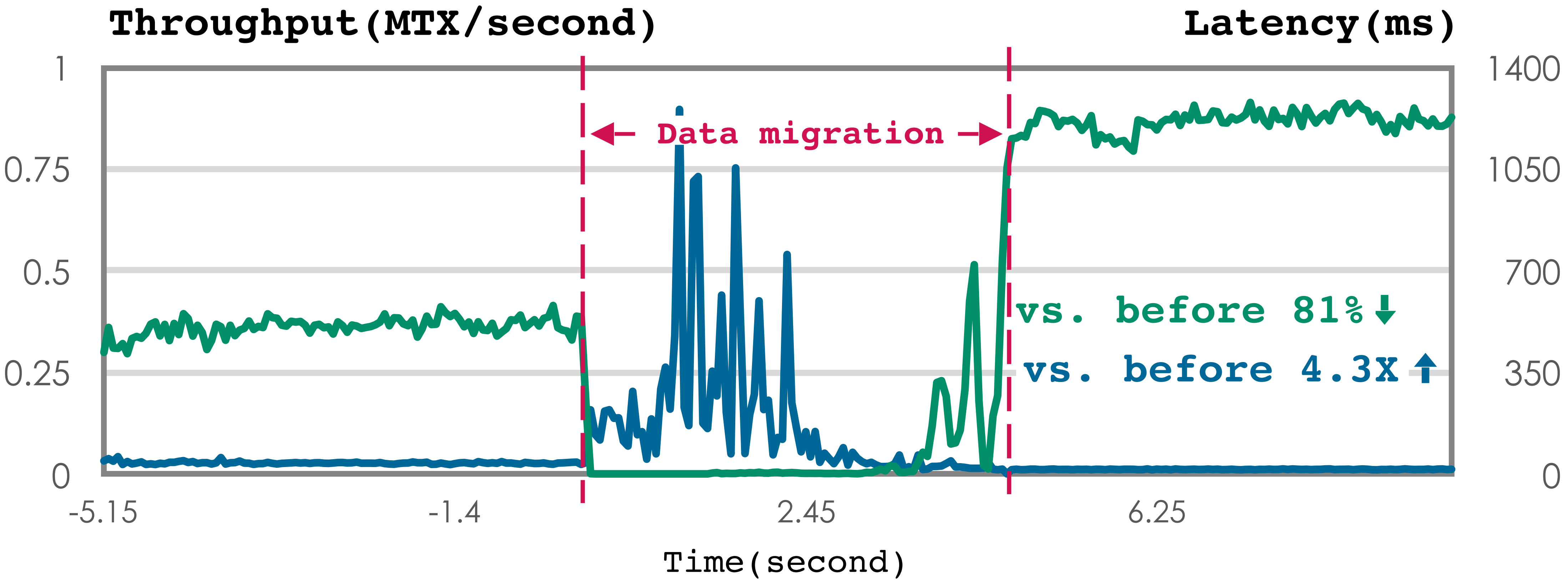
- E-store uses Squall[SIGMOD'15] to migrate data



1. **Blocked** by missing data. (Possibly many times)
2. **Aborted** by migrated data.

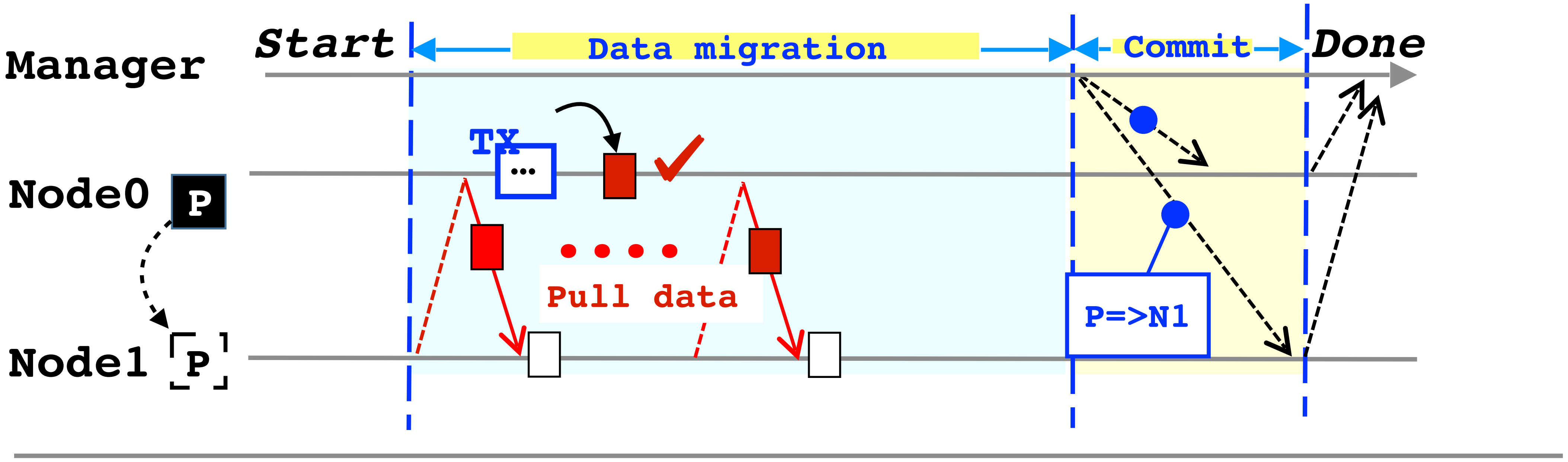
Post-copy is unsuitable for fast TXs

- Using Squall to **balance skewed** TPC-C for **DrTM+R**[Eurosys'16]
  - Due to many affected TXs



SOL#2 Migrating data with Pre-copy

- TXs can **safely** access data at sources

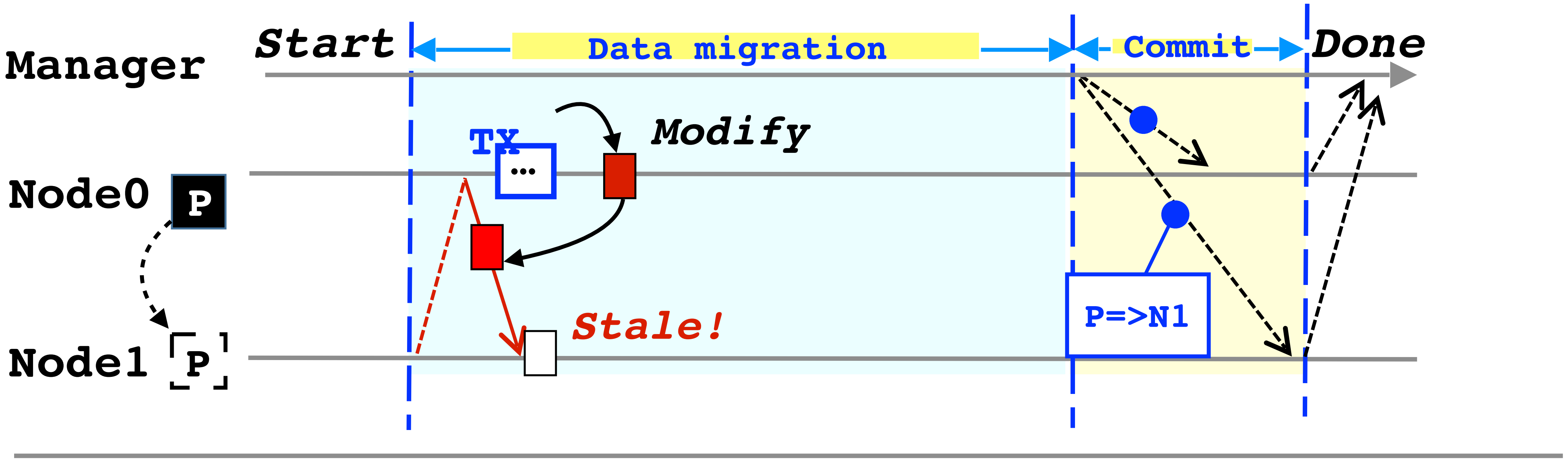


① Migrating **all** data to destination

② Commit the new plan

Pre-copy is not free

- Pre-copy requires **tracking** & **syncing** dirty data

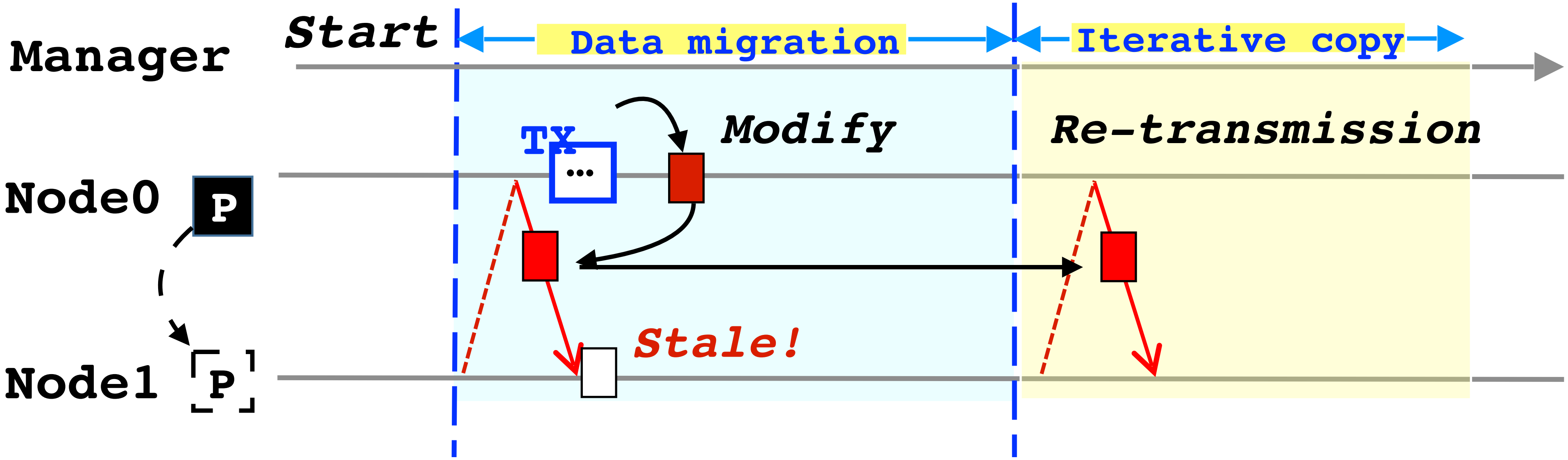


① Migrating **all** data to destination

② Commit the new plan

Pre-copy is not free

- Pre-copy requires **tracking** & **syncing** dirty data



1. **Longer** migration time & **larger** data transmissions
2. TX's tracking overhead

DrTM+B: Fast & Seamless reconfiguration

- Data migration is the most **costly** part
- **Avoids possible data migration** by preferring existing data replicas
- **Pre-copy based approach**: minimizing costs
  - Avoids above shortcomings by leveraging existing fault tolerance mechanisms, i.e logging

Outline

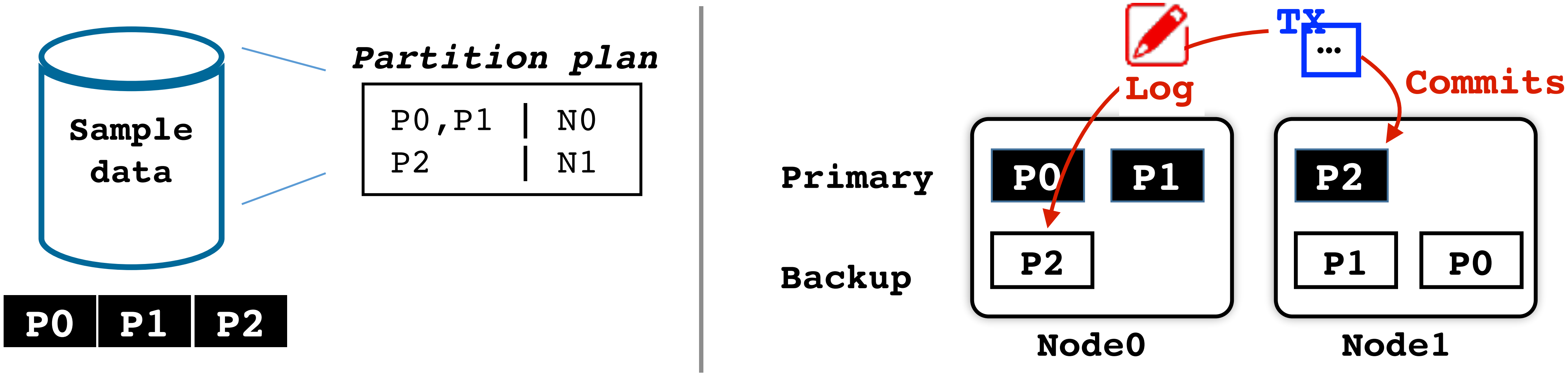
- System architecture
- Reduce data transfer with existing backup
- Data-migration process
- Implementations & Evaluations

Outline

- **System architecture**
- Reduce data transfer with existing backup
- Data-migration process
- Implementations & Evaluations

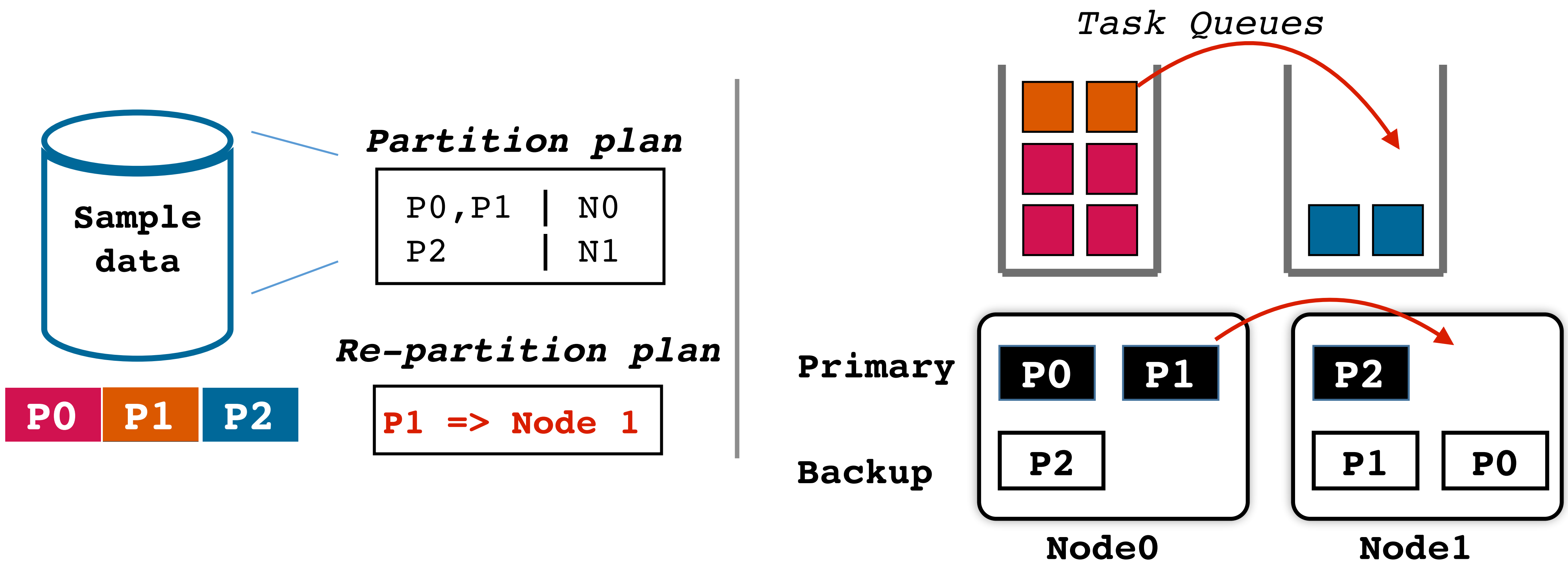
System architecture

- **Sharded** & **replicated** memory store
  - FaSST[OSDI'16], DrTM+R[Eurosys'16],FaRM[SOSP'15]
- Primary-backup synced with **TX's logs**
  - Logs are processed **asynchronously** for efficiency



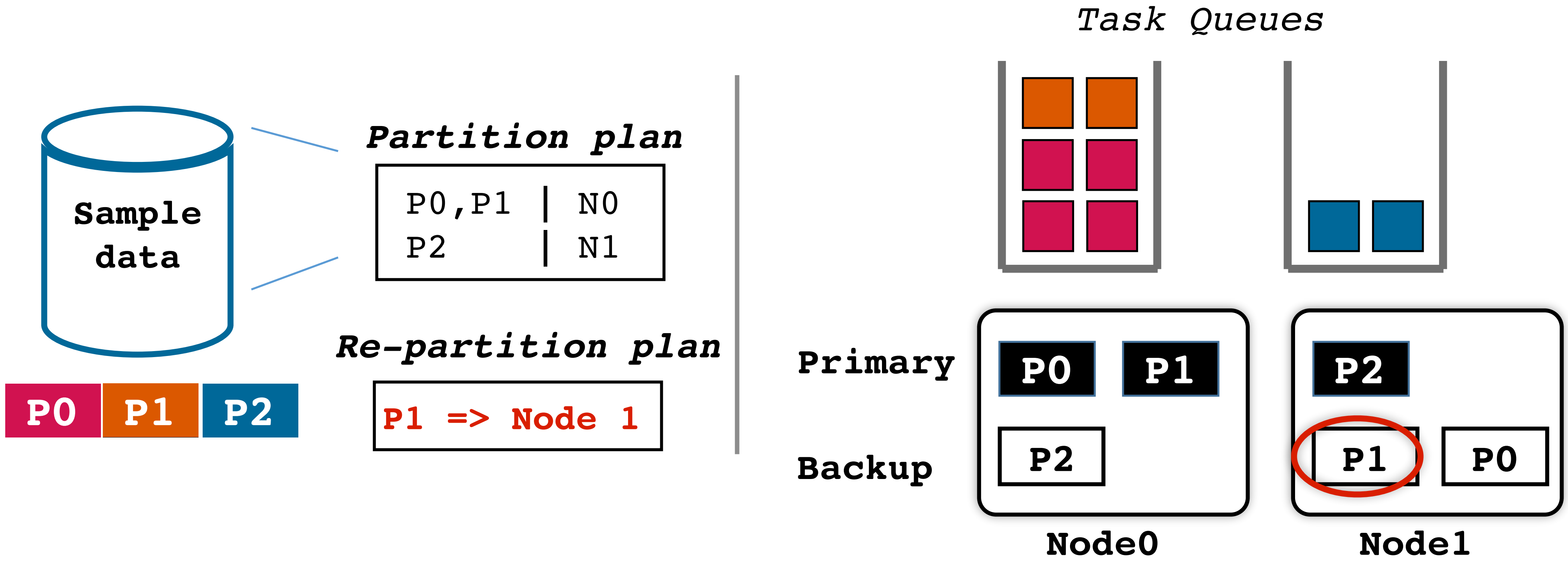
Reduce data transfer with replicas

- R-partition plan assigns **hot data -> cold server**
  - Yet data migration is **costly**



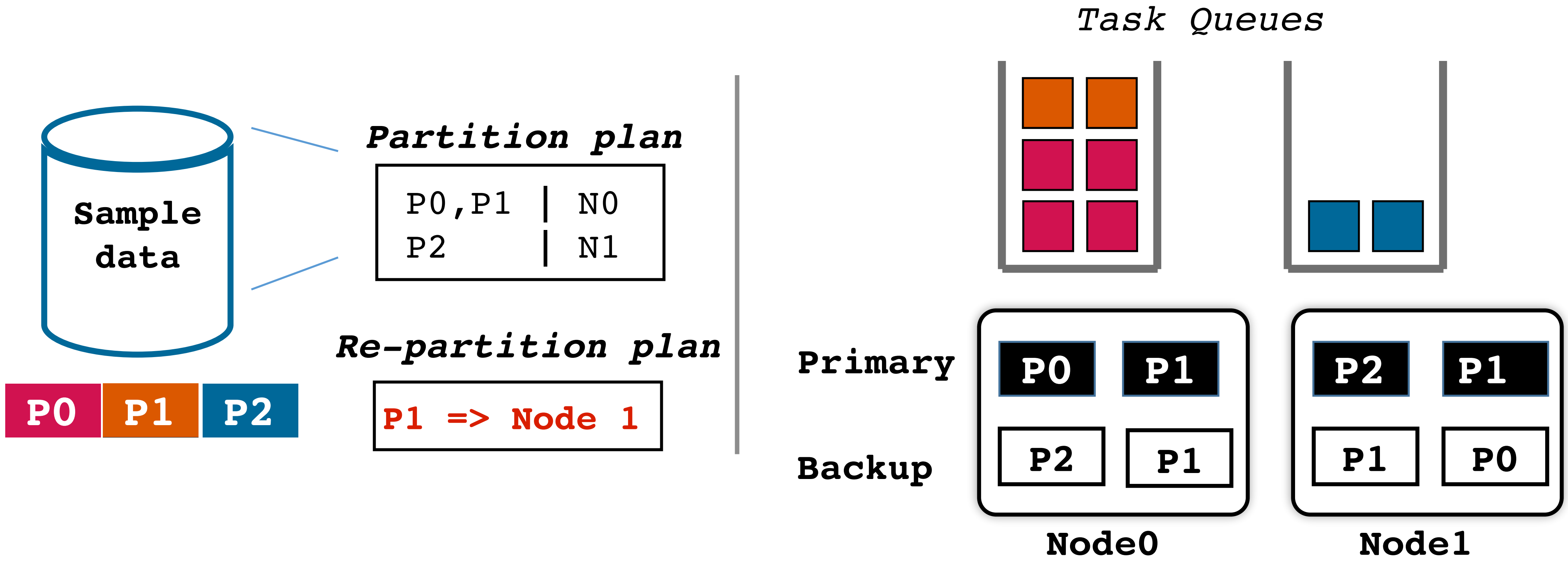
Reduce data transfer with replicas

- R-partition plan assigns **hot data -> cold server**
  - Direct loads to **server with backup data**



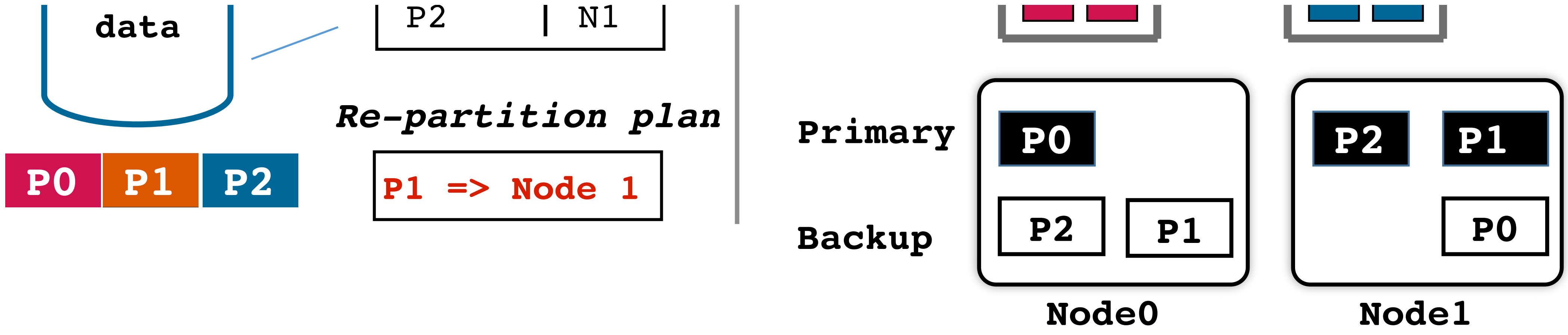
Reduce data transfer with replicas

- R-partition plan assigns **hot data -> cold server**
  - Direct loads to **server with backup data**



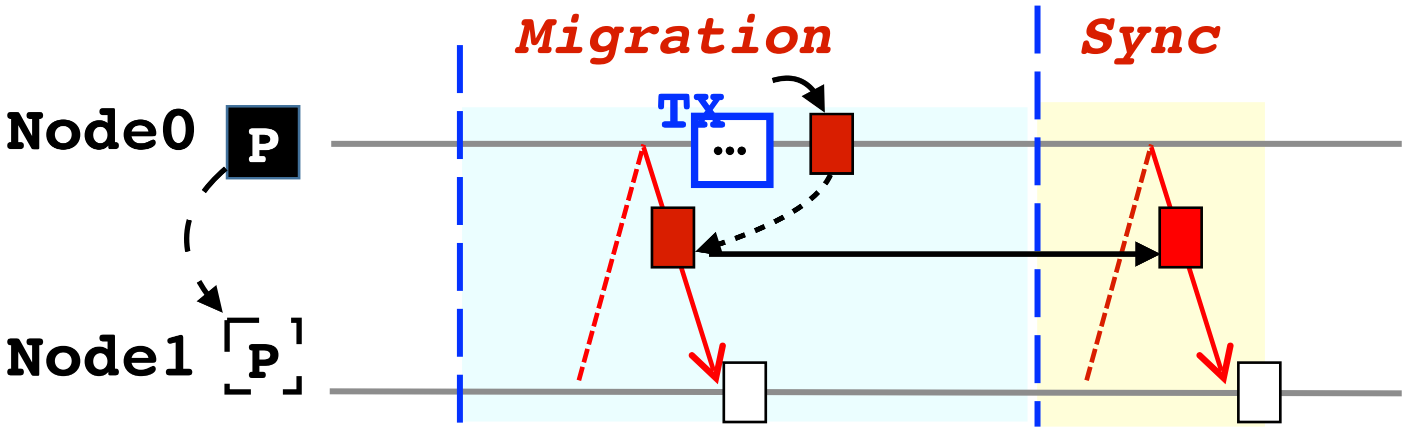
Reduce data transfer with replicas

- Resource is **sufficient** in skewed workloads
  - Direct loads to **server with backup data**
- Data at backup's server does not need migration



Improve pre-copy with log forwarding

- Pre-copy requires **tracking** & **syncing** dirty data during data migration

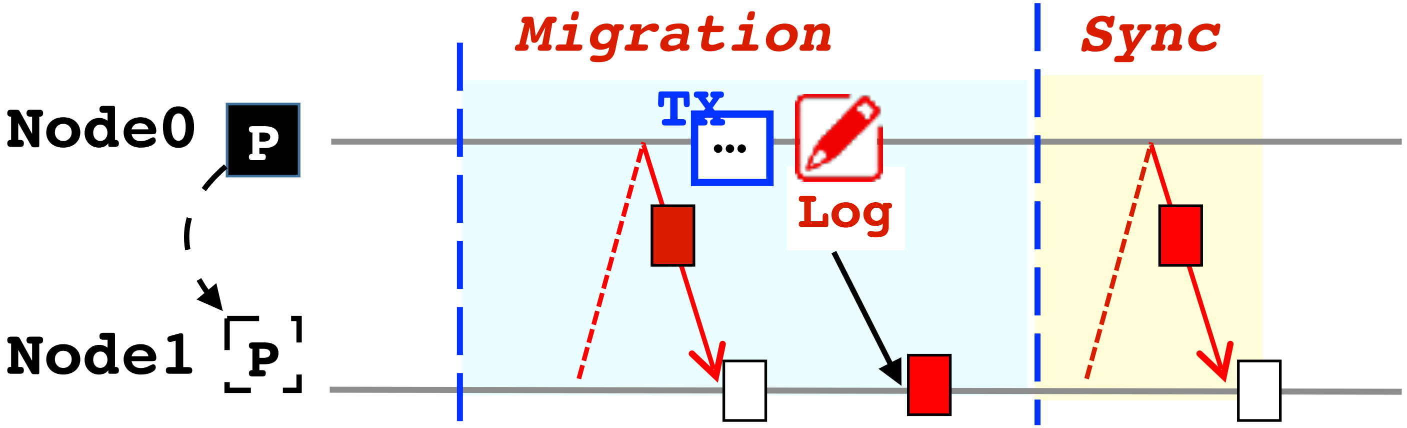


Improve pre-copy with log forwarding

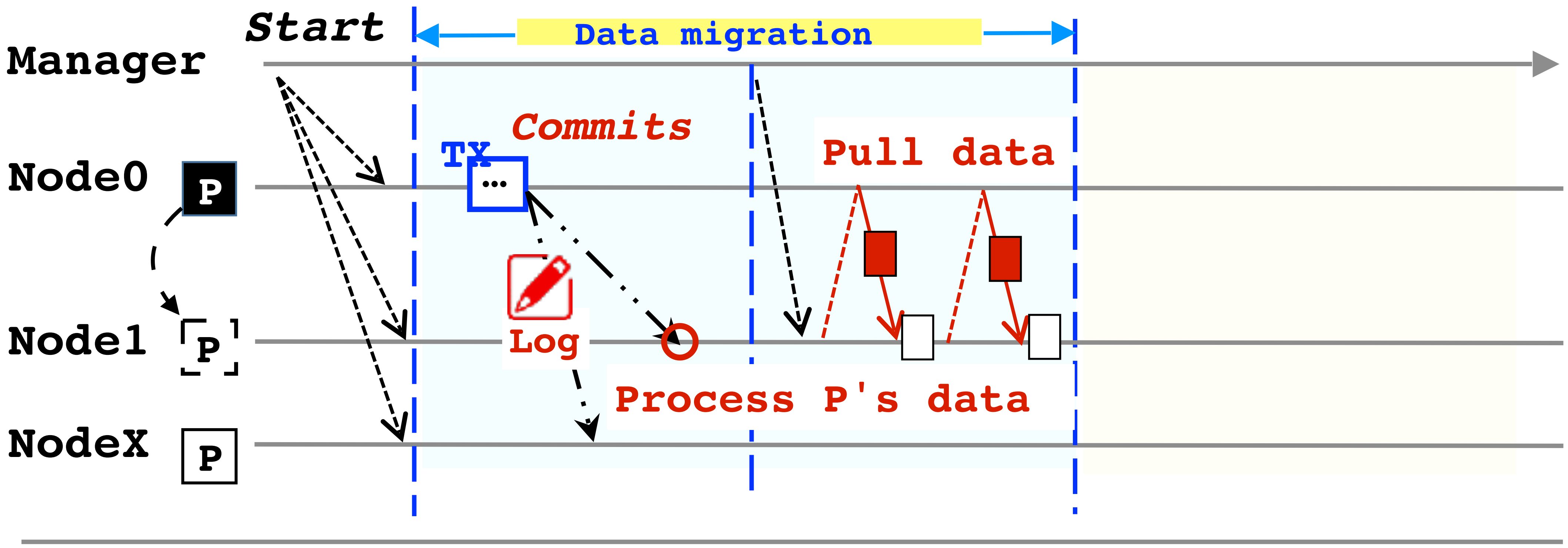
- Pre-copy requires **tracking** & **syncing** dirty data during data migration

  ➡ **Logs** tracks the dirty data

- **Forwards** log **with** data migration



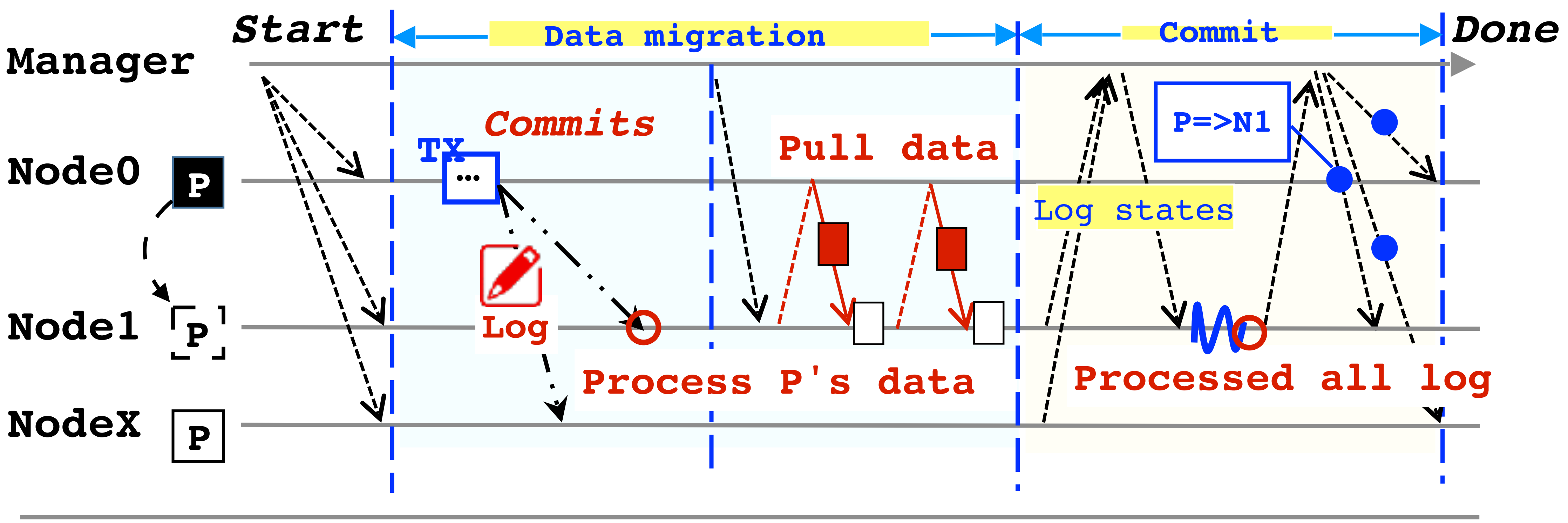
Data migration phase



① Forwards logs to destinations

② Pulls data from sources

Commit phase

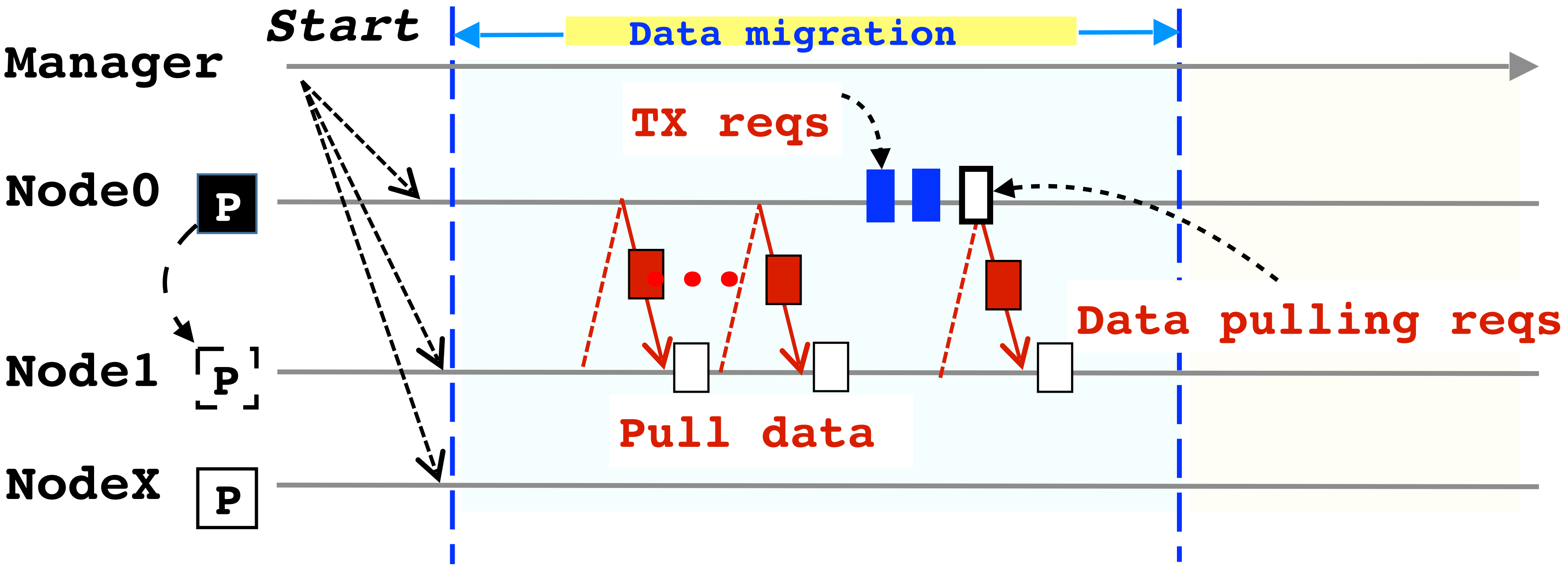


③ Collect log offsets

④ Wait for all **pending** logs to be processed

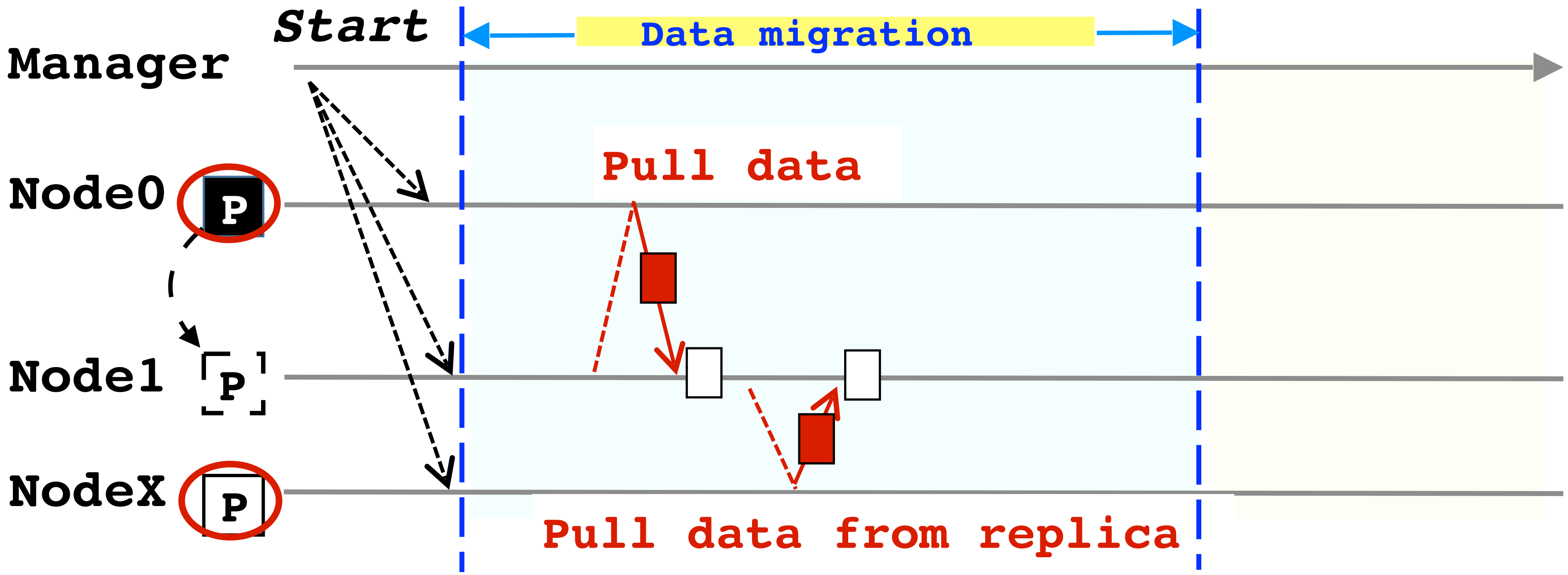
⑤ Commit the new plan

Challenge: Overloaded primary



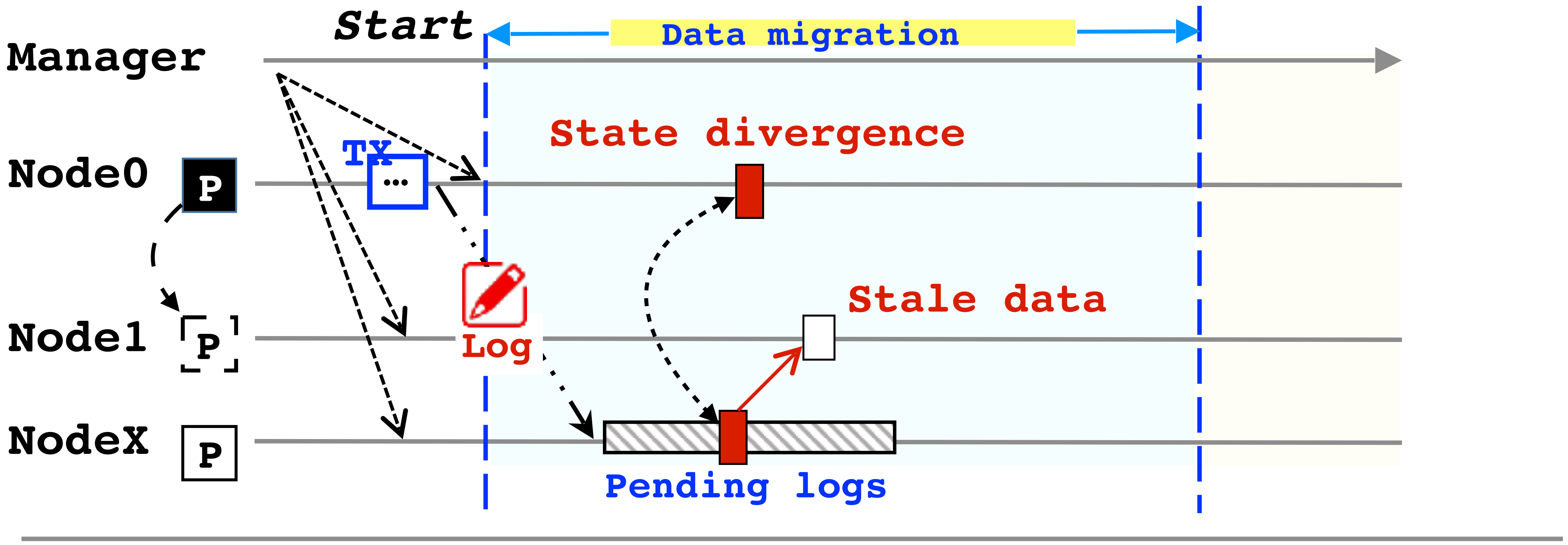
- Primary has become overloaded
  - ➡ Competing CPU resources

Optimization: parallel data fetching



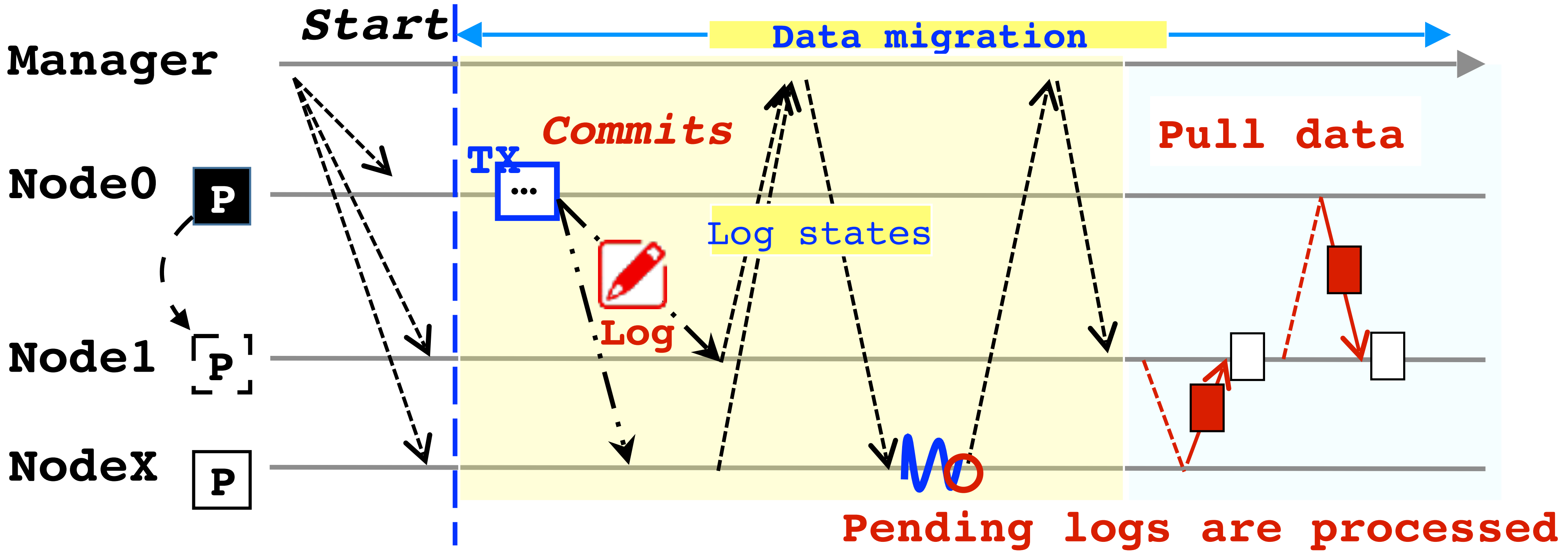
- Parallel fetching data from **all** replicas
  - ➡ As backup contains **nearly the same** content as primary

Challenge: Stale backup



- Logs are **asynchronously** processed at backups
  - Directly fetching from backup causes inconsistency

Pre-sync before parallel data fetching



① Forwards logs

② Syncs log states

③ Wait for pending logs

④ Parallel data fetching

Other Specific Implementation

- Based on DrTM+R[Eurosys'16]
- Cooperative commit protocol
- Replication-aware planner
- Workload monitor
- Fault tolerance

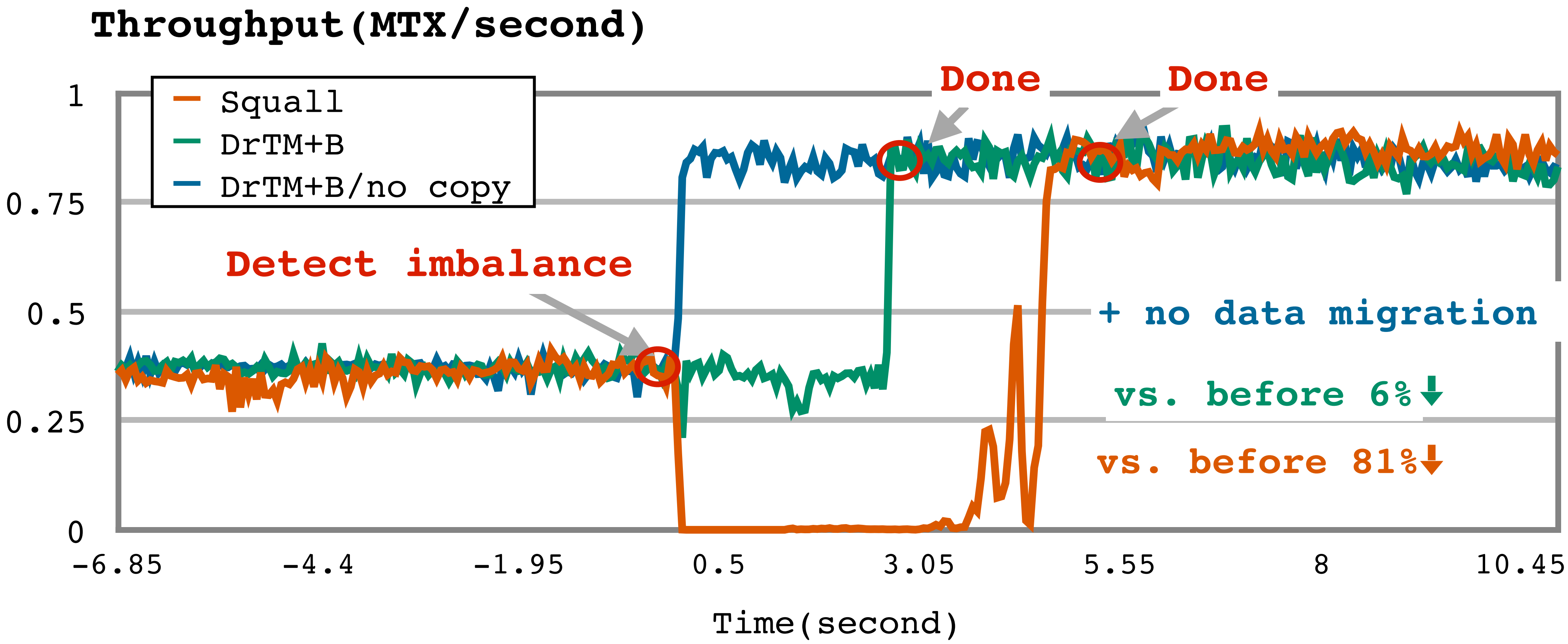
Evaluations

- Platform:
  - ➡ 6-node local cluster
  - ➡ 3-way replication enabled
- Benchmarks:
  - ➡ TPCC & Smallbank with 2 skewed settings[1]
- Comparison
  - ➡ Squall[SIGMOD'15] on DrTM+R

[1] **Low:** 60% accesses goes to 1/3 warehouse
High: 40% accesses goes to 4 warehouses on one node, the rest is the same as low skew

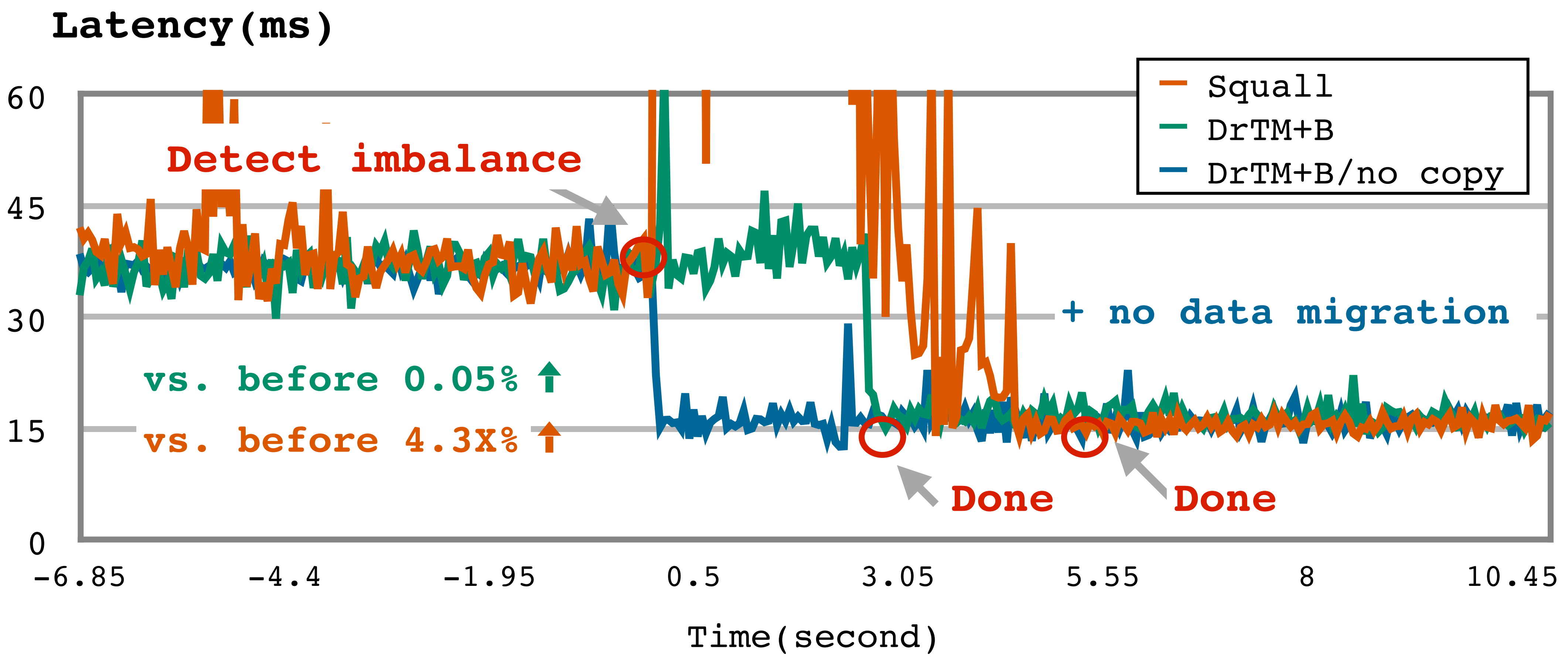
Performance of load balance

- Reconfiguring TPC-C with low skew



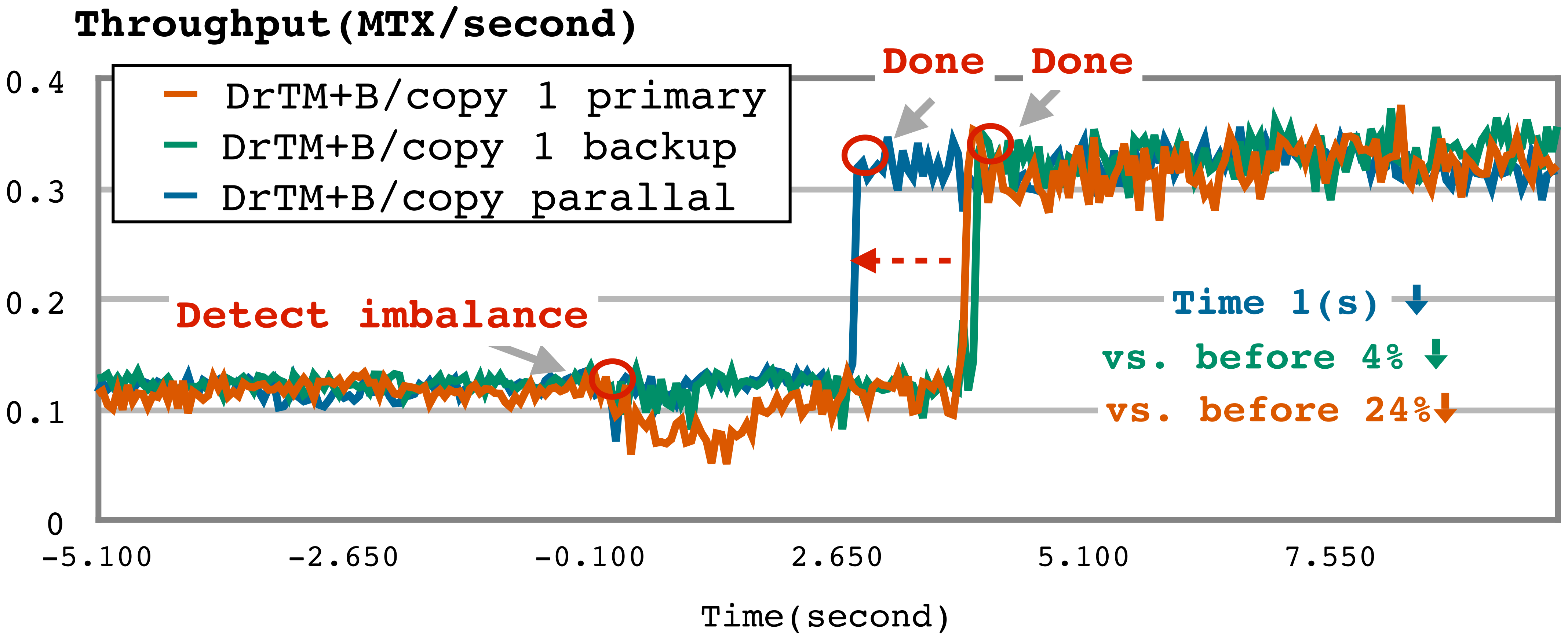
Performance of load balance

- Reconfiguring TPC-C with low skew



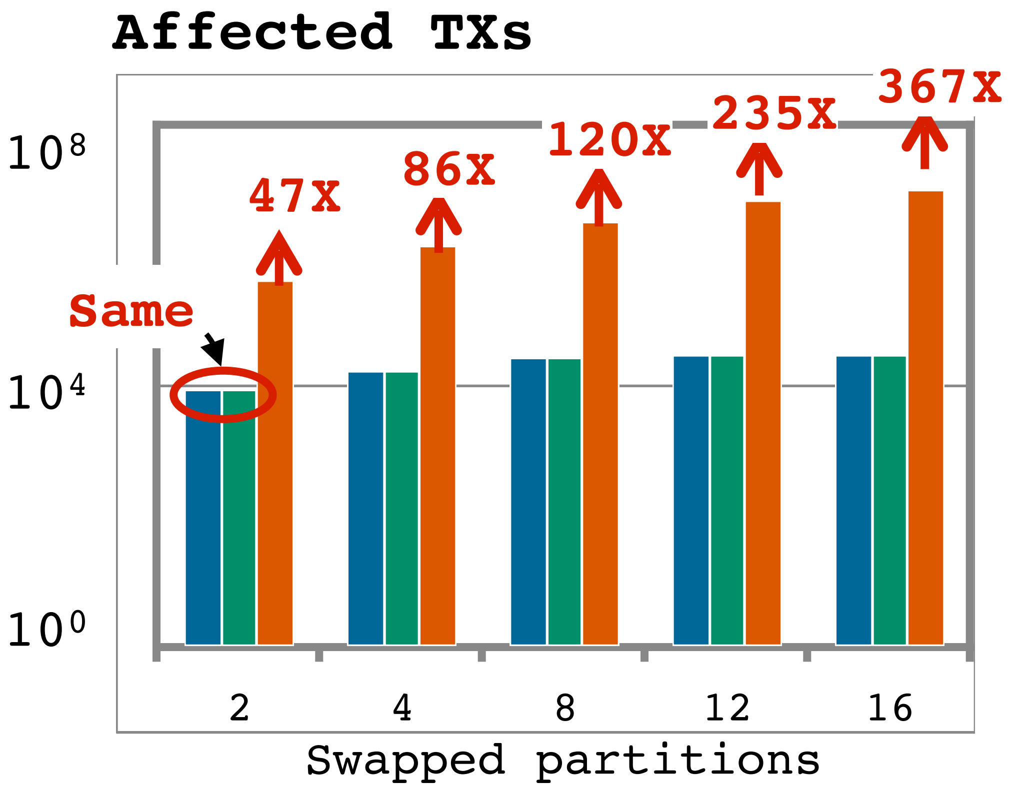
Breakdown of data migration

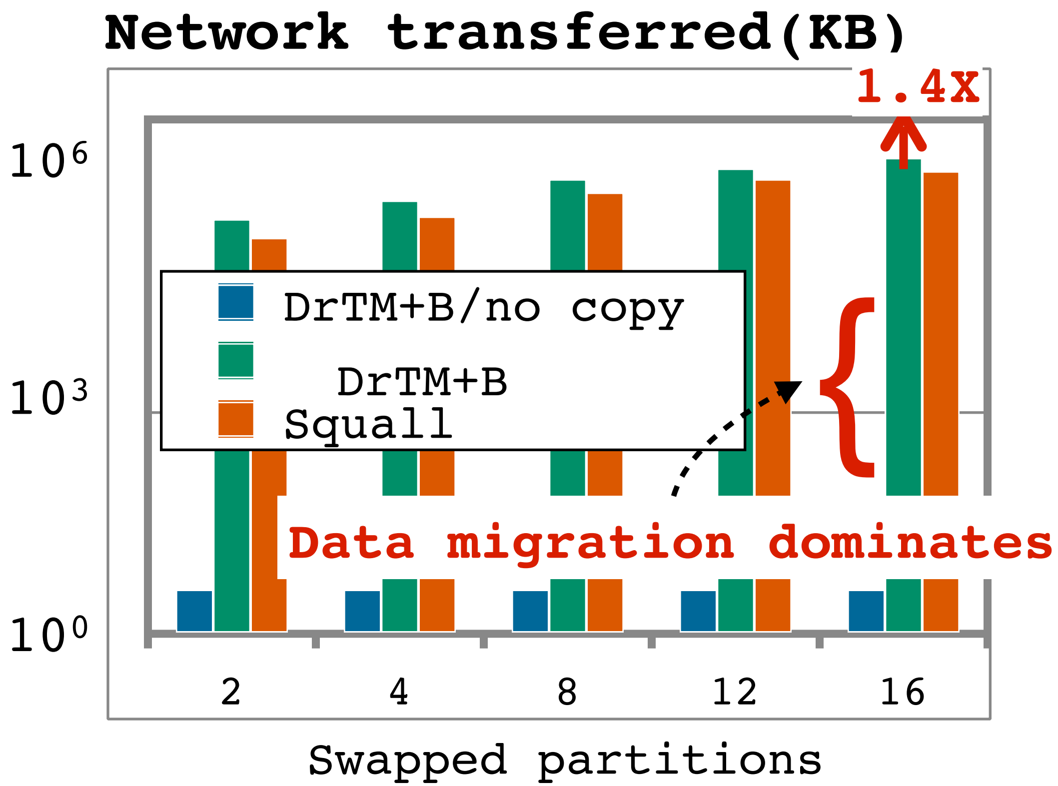
- Reconfiguring TPC-C with high skew



Affected TXs & Network transferred

- Micro-benchmark based on TPC-C
  - Swapping partitions between 2 nodes





Conclusion

- Real workloads are **dynamic** & **skewed**
  - Requires fast & seamless live reconfiguration
- DrTM+b provides fast live reconfiguration
  - **Optimized with features** in transactional systems
  - Nearly **no-effect** to TXs

Thanks & Questions?