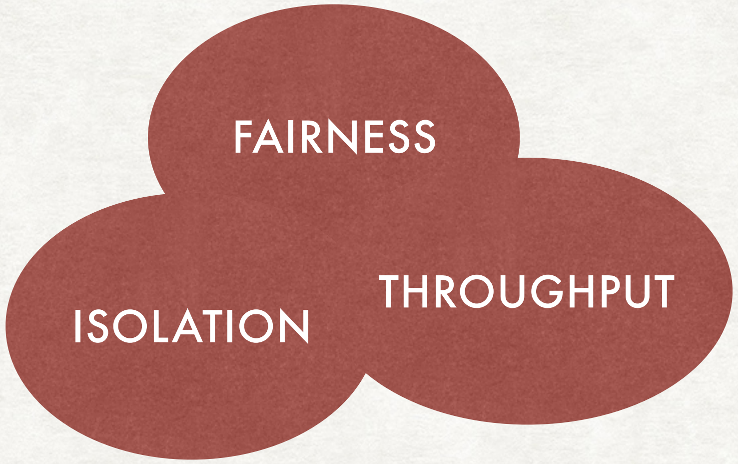
FIT: A Distributed Database Performance Tradeoff

Jose M. Faleiro, Yale University

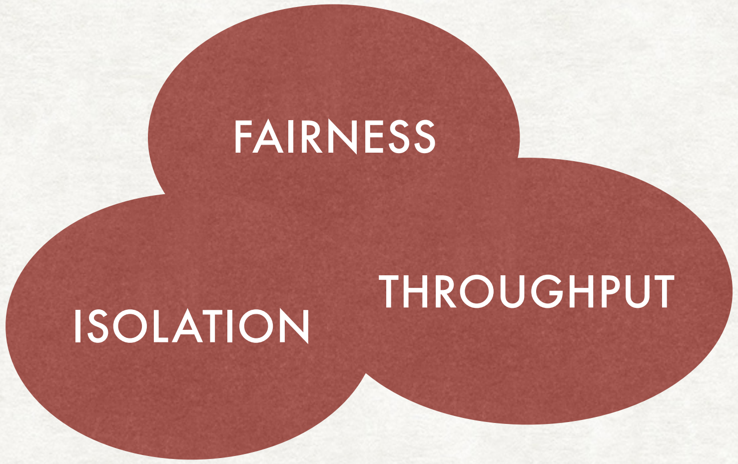
Daniel J. Abadi, Yale University

Presented by Bojun Wang



Isolation v.s. Throughput **and Fairness**

- Strong isolation —> poor throughput
- poor isolation —> good throughput
- But fairness is another factor: **FIT 3-way trade-off**



DEFINITIONS

- Distributed Transaction: reads/writes involves records from multiple partitions
- ASSUMPTION: a distributed database must satisfy Liveness, Atomicity, and Safety

DEFINITIONS

- Liveness: If distributed transaction is always re-submitted whenever it sees a system-induced abort, it’s guaranteed to commit eventually.
    - system-induced abort: caused by partition failure or deadlocks
    - logic-induced abort: caused by logic inside transaction
- Safety: all nodes involved in a distributed transaction must all agree to commit, otherwise abort.
- Atomicity: all/none updates of a transaction are in database.

Fairness (intuitively)

- Database system does not deliberately prioritize nor delay certain transactions.
  - Never artificially adds latency to a transaction for the purpose of facilitating the execution of other transactions.

UNFAIRNESS EXAMPLES

- Example 1: “group commit”
  - writing logs to disk is slow
  - write N transactions’ logs in batch, single disk write
  - better overall throughput
  - but some transactions cannot commit until threshold N is met
- Example 2: “lazy evaluation”
  - collect transactions that reads/writes spatial close records
  - defer execution
  - amortize cost of bring records into memory
  - but some transactions have to wait for other transactions

DEFINITIONS

- Synchronization Independence: One transaction cannot cause another transaction to block or abort. (Even with conflicting data accesses)
- Synchronization Independence implies Weak Isolation
    - running with synchronization independence, cannot guarantee any form of isolation

FIT TRADEOFF

- a distributed transaction needs **coordination** between partitions
- Strong isolation

—> conflicting transactions must wait

— > coordination increases wait time

—> bad throughput

FIT TRADEOFF

- Distributed Transaction needs coordination between nodes
- ~~Strong isolation~~

    —> conflicting transactions must wait synchronization independence

        — > coordination increases wait time reduce impact of coordination

Weak Isolation **Good Throughput**

FIT TRADEOFF

- Strong Isolation
  - coordination makes conflicting transaction wait longer
  - But giving up Fairness can reduct this impact
- Example
  - Do coordination outside of transaction
    - Thus not increasing conflicting transactions wait time
- **Better Throughput** **Bad Fairness**

FIT IN EXAMPLES

	Fairness	Isolation	Throughput
G-Store		✓	✓
Calvin		✓	✓
Spanner	✓	✓	
Cassandra	✓		✓
RAMP	✓		✓

EXAMPLES

G-Store

Isolation Throughput Fairness

- KeyGroup
  - Put a set of keys into one 'leader' partition
  - Reduce coordination cost
  - Not fair to keys not in KeyGroup
  - Some Transactions delayed to form new KeyGroup

EXAMPLES

Calvin

Isolation Throughput Fairness

- Pre-process a **batch** of transactions
  - generate total ordering, i.e. a redo log
  - serializable isolation level
  - eliminate deadlock; avoid expensive planning for failures
    ~~forced-log writes, synchronous replication~~
  - minimize coordination cost
  - Pre-process a large batch of transactions for throughput
    Unfairness

EXAMPLES

Spanner

Isolation Throughput Fairness

- Serializable Isolation level
- Guarantee Fairness
- 2-phase-commit in replicated setting
    - synchronously replicate every node's prepare vote
    - synchronously replicate coordinator's final commit decision
- Coordination during transaction —> hurt throughput

EXAMPLES

Cassandra

Isolation Throughput Fairness

- “batch transaction”: UPDATE SET DELETE
    - allow clients to see partial results
    - give up isolation
    - no coordination required for conflicting “transactions”
    - good throughput and good fairness

EXAMPLES

RAMP

Isolation Throughput Fairness

- Read Atomic: All/None of a transaction updates are visible
- Implemented by Read Atomic Multi-Partition
  - guarantee synchronization independence
  - weak isolation

FIT IN EXAMPLES

	Fairness	Isolation	Throughput
G-Store		✓	✓
Calvin		✓	✓
Spanner	✓	✓	
Cassandra	✓		✓
RAMP	✓		✓

FIT, IN MULTICORE DATABASE

Isolation Throughput Fairness

- SILO: Multicore Machine Database, Serializable
- Tradeoff fairness to gain throughput
    - append logs to shared in-memory buffer
    - expensive to append logs due to synchronization cost
    - each core store logs in core-local buffer
    - periodically move logs from local to shared
- Amortize synchronization cost over batch of transactions. Unfairness

FIT, IN MULTICORE DATABASE

Isolation Throughput Fairness

- Dopple: Multicore Machine Database, Serializable
- joined phase ——aggregate—— split phase
- joined phase, only one record exists, all transaction allowed
- split phase, replica, only allow commuting operations. Unfairness

FIT TRADEOFF

Coordination is a price

- Pay it during transaction + strong isolation ==> poor throughput
- Pay it before transaction + strong isolation ==> unfairness
- Give up isolation (reduces coordination impact) ==> fairness & throughput