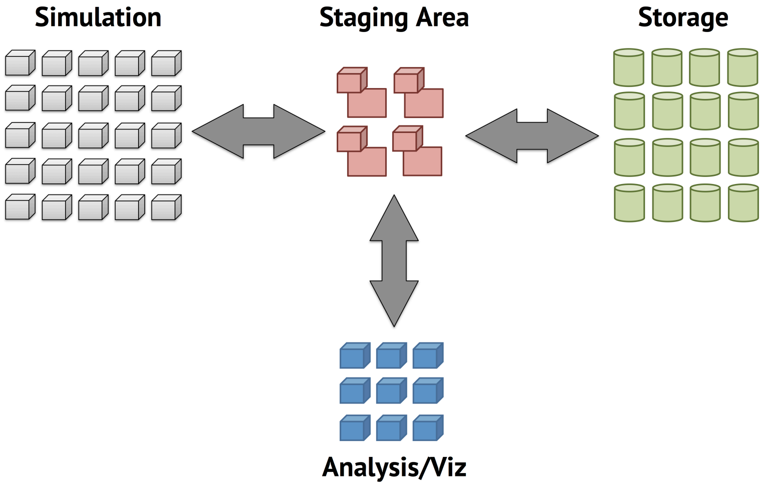
Exploring Trade-offs in Transactional Parallel Data Movement

Ivo Jimenez, Carlos Maltzahn (UCSC)

Jay Lofstead (Sandia National Labs)

November 18, 2013

The need for Transactional Atomicity



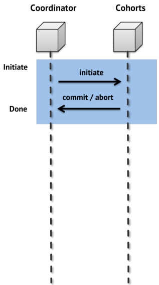
The difference with Databases

- In terms of ACID, we want:
  - **A**tomicity
  - **D**urability
  - Leave **I**solation/**C**onsistency to the clients
- Single Transaction (vs. thousands)
- Massive amount of cohorts (vs. hundreds)

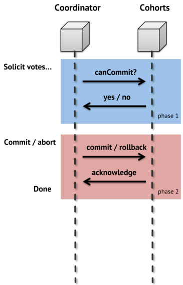
The approach

- Assume that storage servers can do:
    - multi-version concurrency control
    - per-object visibility control
- Clients handle consensus

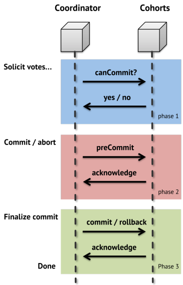
Consensus Protocols



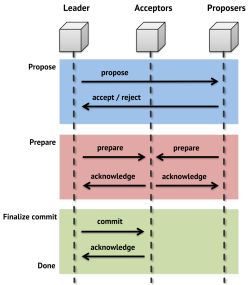
NBTA



2PC



3PC



Paxos Commit

NBTA

- **N**on-**b**locking **T**ransactional **A**tomicity
- “HAT” formalization (Bailis et al. VLDB 2014)
- In the context of Highly-available systems
- Can also be applied in synchronous systems to achieve very low overhead

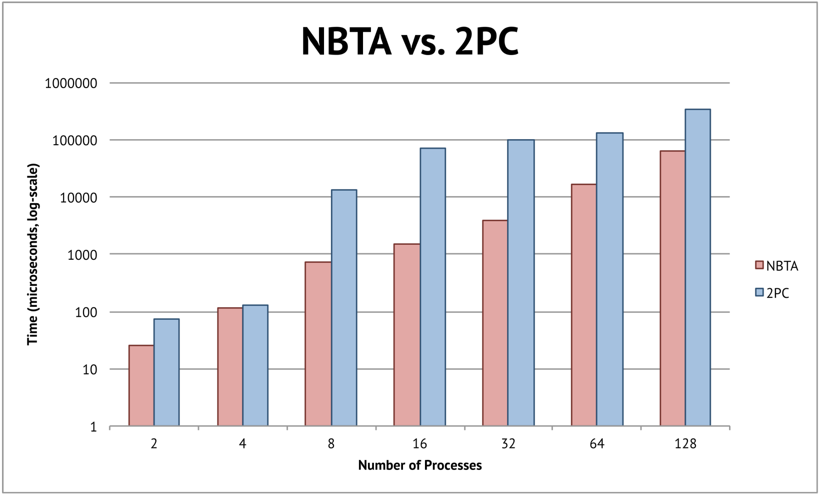
Features

Protocol	Fault Model	Block	Async	Replication
NBTA	none	Yes	No	No
2PC	fail-stop	Yes	No	No
3PC	fail-stop	No	No	No
Paxos	fail-recover	No	Yes	Yes

Our goal

- One-size-fits-all solution won’t work
- Let users pick based on their needs:
  - Length of job
  - MTTF
  - fault modes
  - etc
- We want to explore trade-offs and characterize protocols based on the user needs

Preliminary Evaluation



Future Work

- Incorporate fault-tolerance
    - Cohort failure: can recover individually
    - Coordinator failure: 3PC, Paxos
- Coordinate asynchronously
    - No need to wait for global consensus

Related Work

- DOE’s Fast Forward Storage and I/O. The FastForward approach is similar to the NBTA protocol.
- Fault-tolerant MPI make use of consensus protocols to identify faulty processes.
- Recovery in multi-level checkpoint restart.

Thanks!