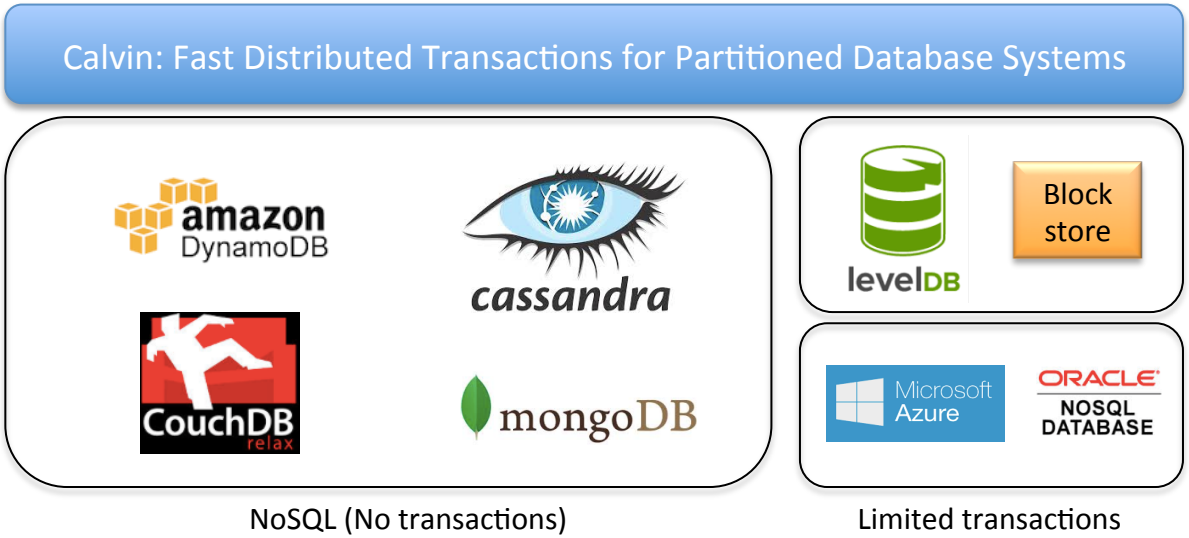
Scalability in Calvin

Discussant: Yeounoh chung

yeounoh_chung at brown

Calvin is amazing!

- Calvin transforms a non-transactional storage system into a shared-nothing, *near-linearly scalable* database system with *full distributed ACID transactions*.



Magic trick: removes synchronization overhead from the distributed system! How? Use deterministic global transaction ordering.

Linear scalability at what cost?

- High scalability is achieved by removing the concurrency control overhead. Is this OK?
  - First, assumption: very high-contention workload [1].
  - Now, all conflicting execution is in serial order [1, 2].
    - Long transaction is a problem!
    - Dependent transaction is a headache!

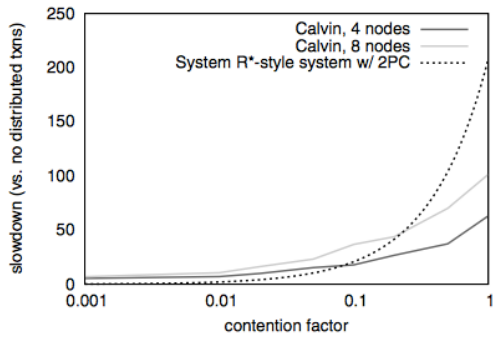
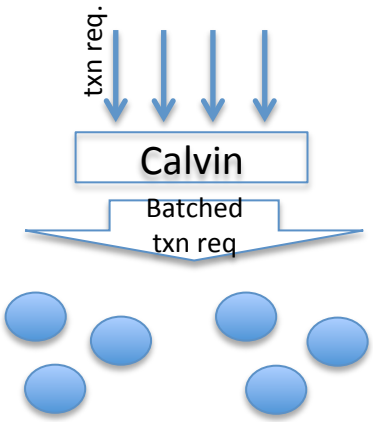


Figure 6: Slowdown for 100% multipartition workloads, varying contention index.



$U(x)$:
  $y \leftarrow \text{read}(x)$
  $\text{write}(y)$

$U_1(x)$:
  $y \leftarrow \text{read}(x)$
  $\text{newtxnrequest}(U_2(x, y))$

$U_2(x, y)$:
  $y' \leftarrow \text{read}(x)$
  if $(y' \neq y)$
    $\text{newtxnrequest}(U_2(x, y'))$
    $\text{abort}()$
  else
    $\text{write}(y)$

[1] Thompson et al. "Calvin: Fast Distributed Transactions for Partitioned Database Systems," SIGMOD '12
[2] Thompson et al. " The Case for Determinism in Database Systems," VLDB '10

Conclusion

- Calvin is amazing if the workload is *not embarrassingly partitionable* and with *high contention*.
- Otherwise, Calvin can add overheads.
  - Long transactions, dependent transactions.
  - Most real workload in which concurrency control is not the big impediment for the scalability.