Microsoft Research

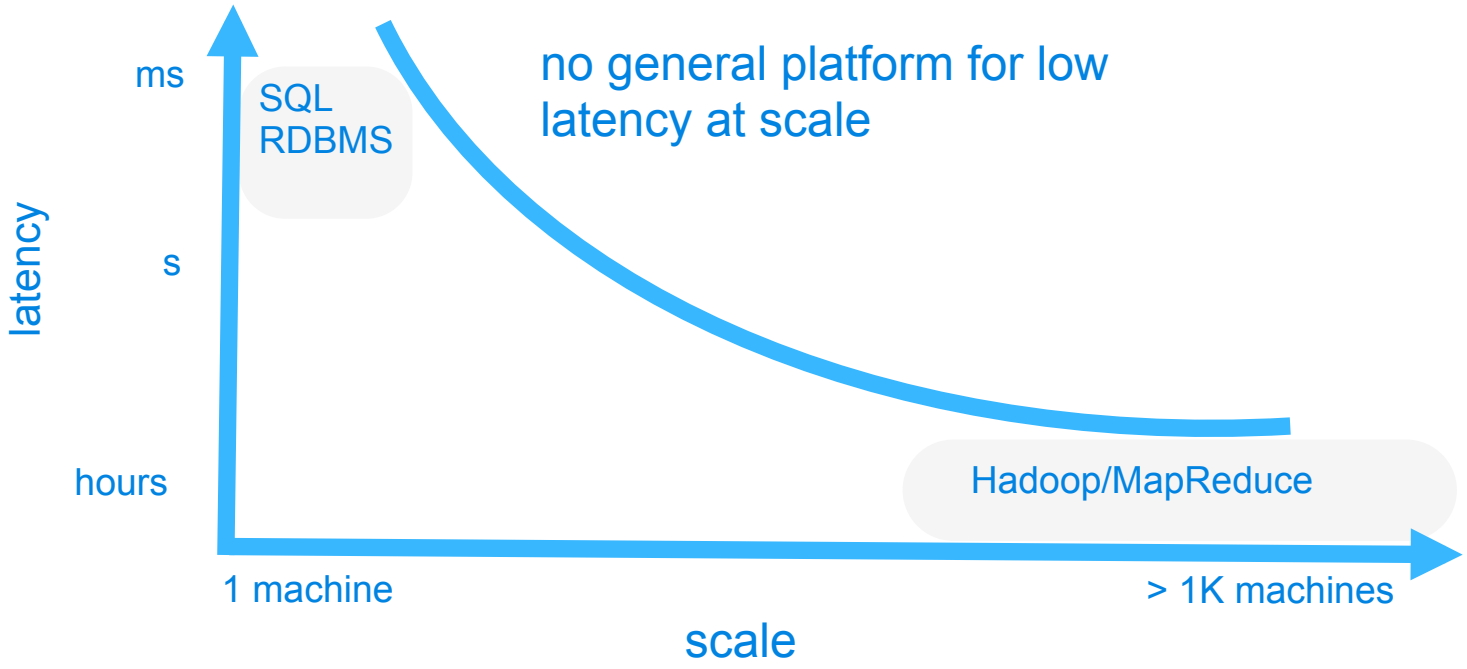
Microsoft

A1 and FARM

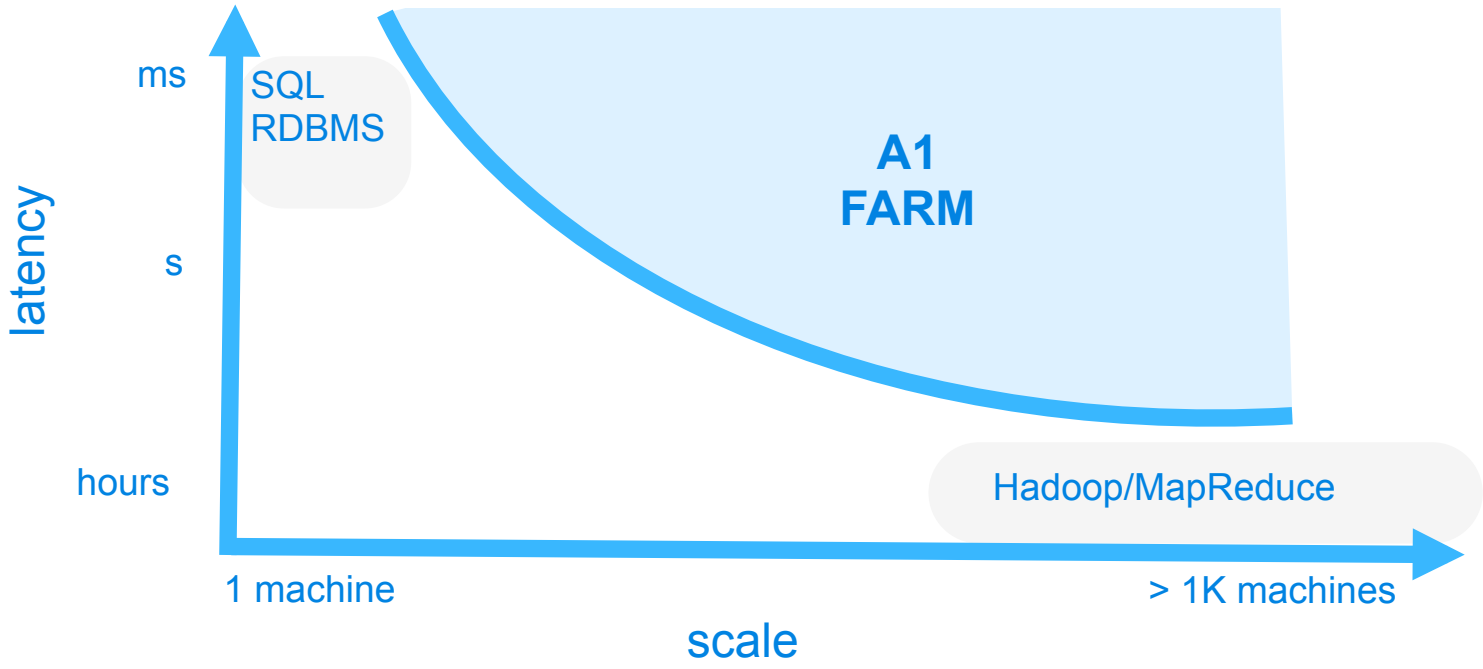
scalable graph database on top of a transactional memory layer

Miguel Castro, Aleksandar Dragojević, Dushyanth Narayanan,
Ed Nightingale, Alex Shamis, Richie Khanna,
Matt Renzelmann, Tim Tan,
Chiranjeeb Buragohain,
Knut Magne Risvik

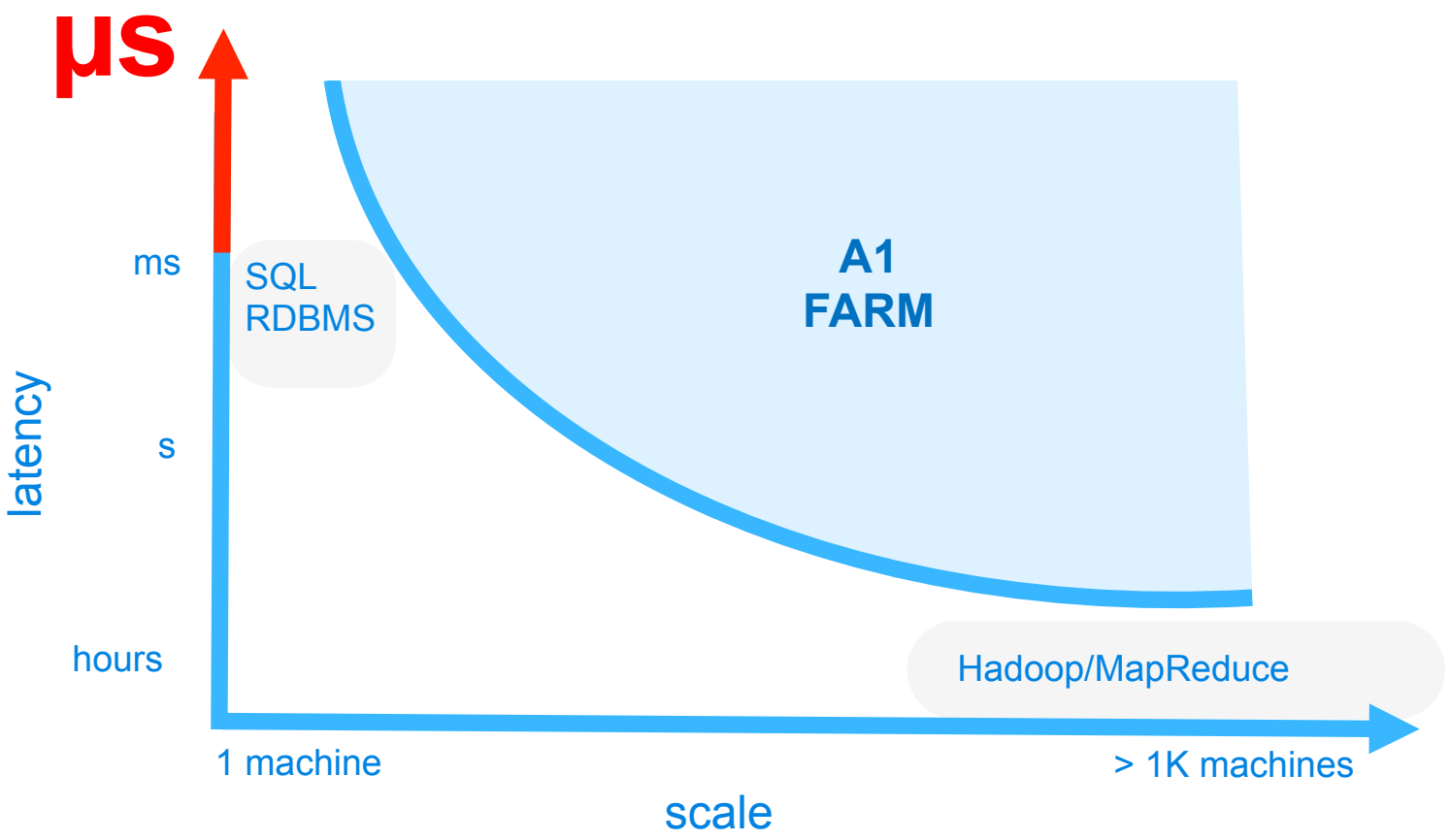
Distributed computing today



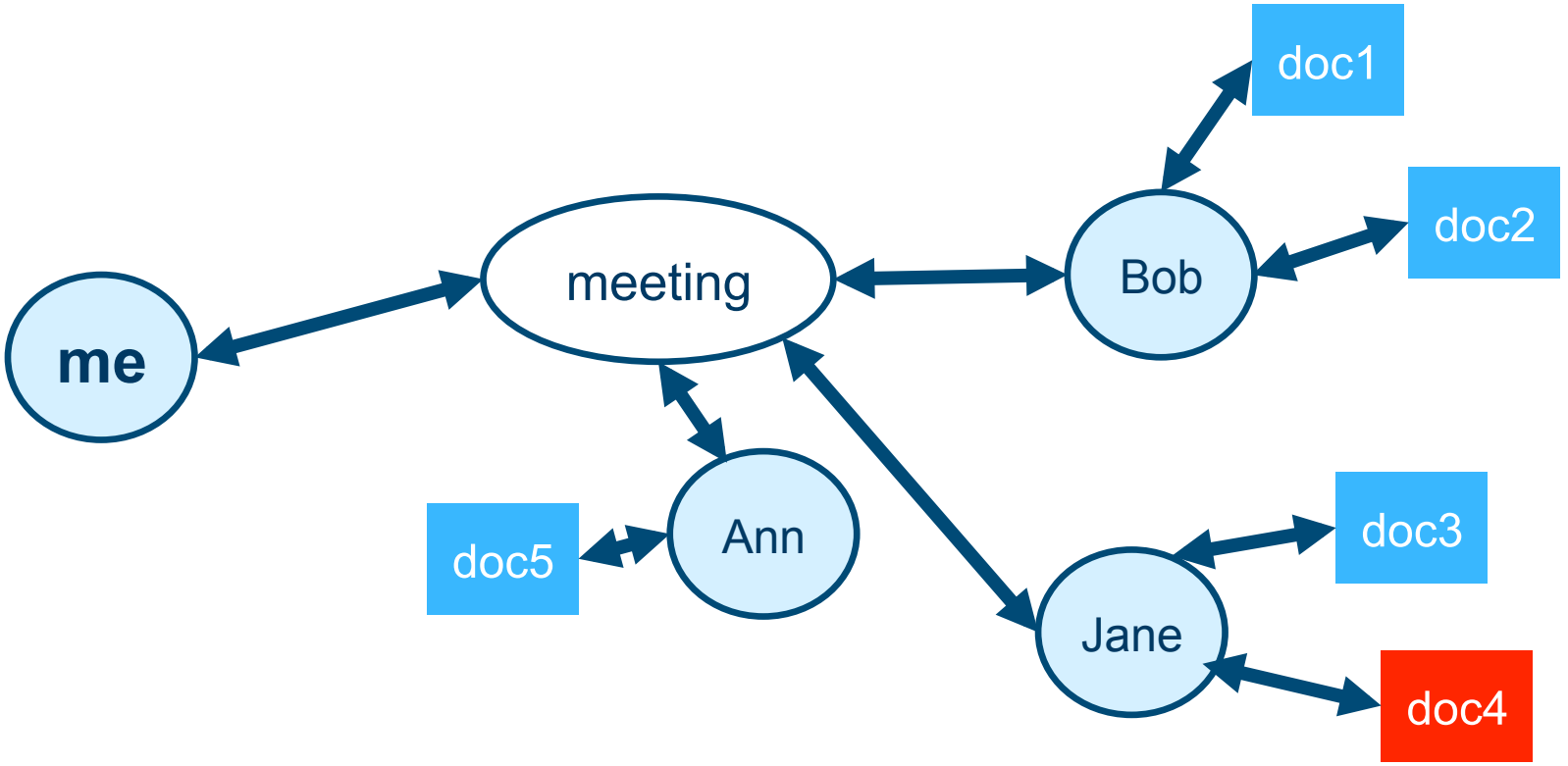
A platform for low latency computing



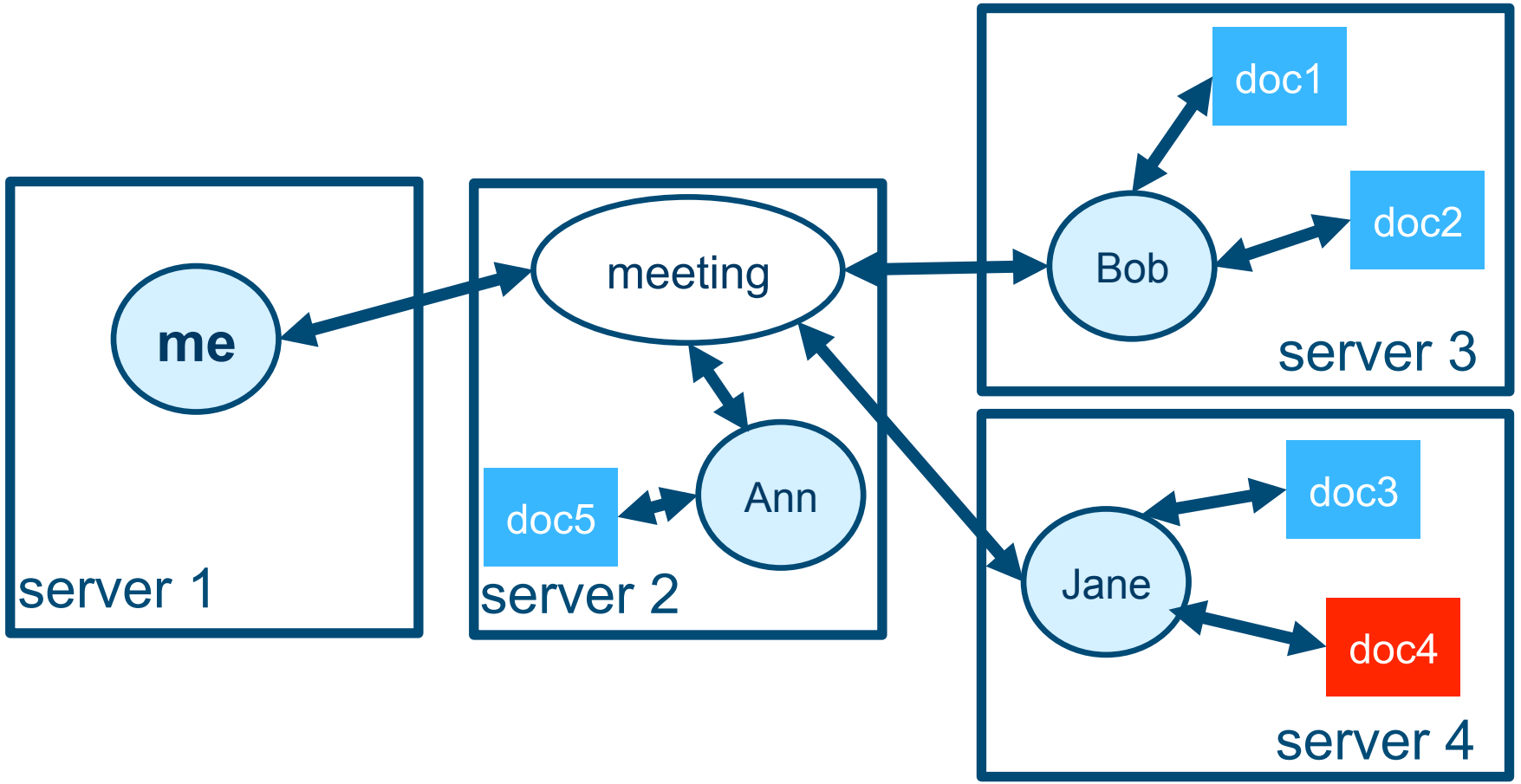
A new frontier for low latency computing



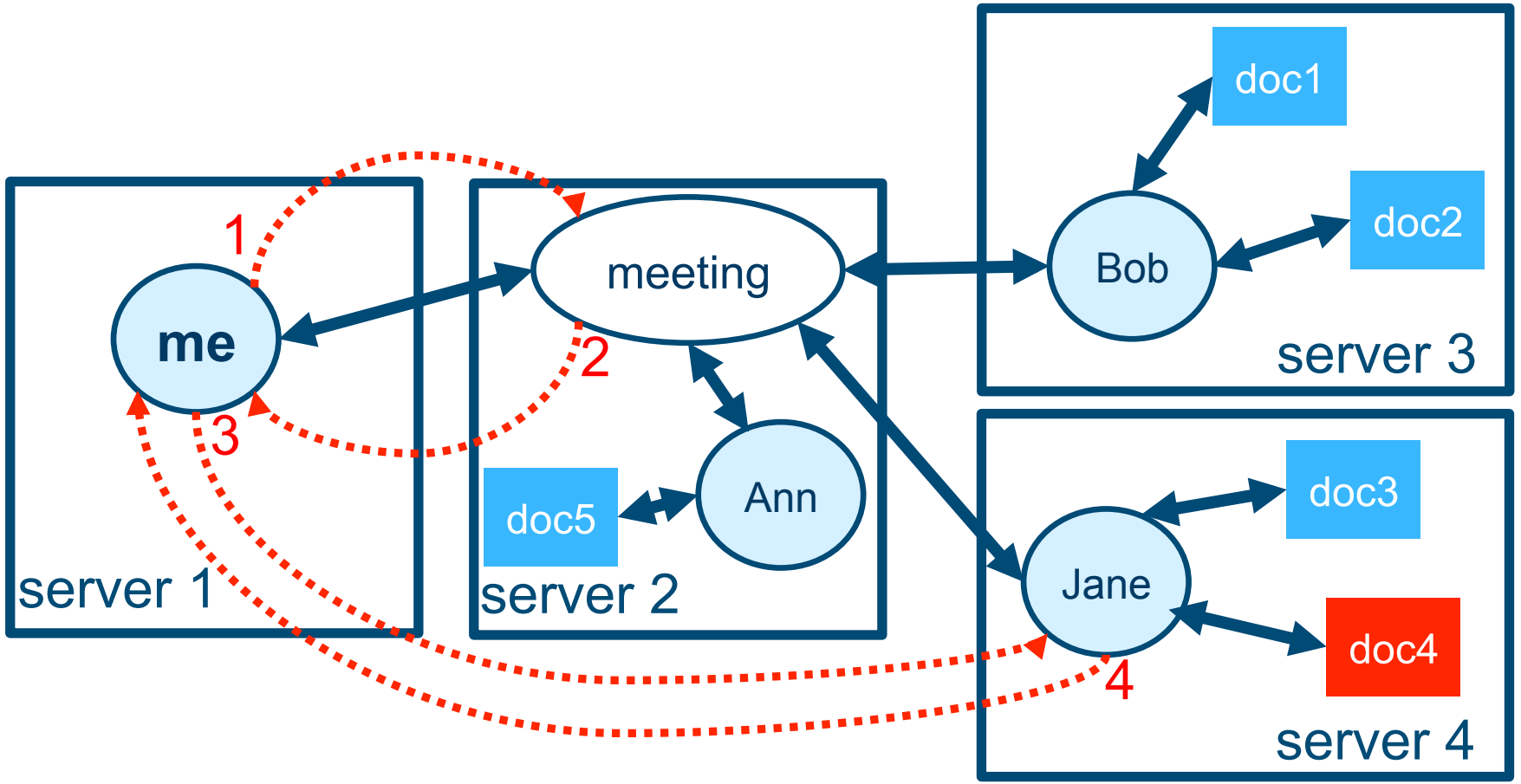
A graph workload



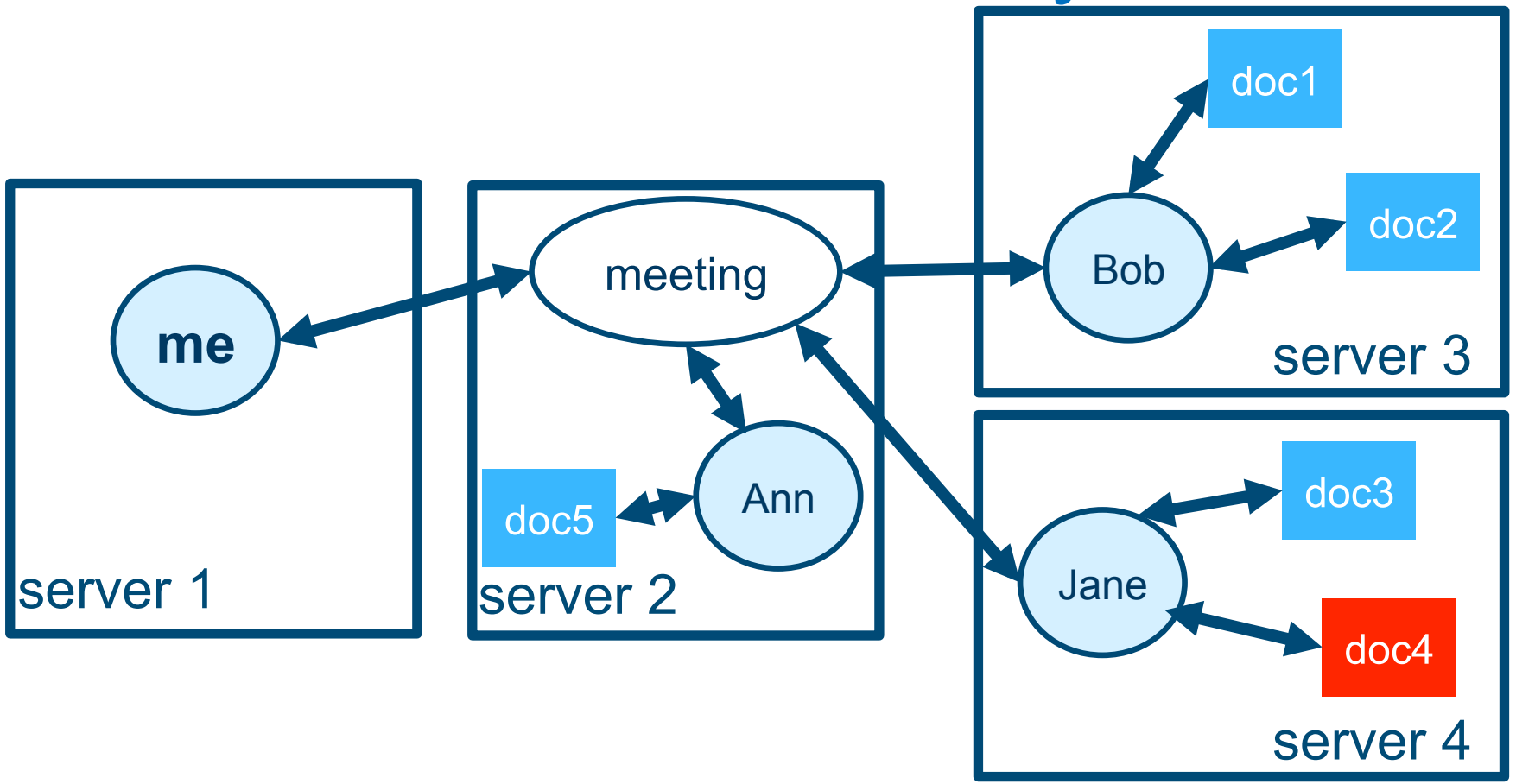
Scale out


me
server 1
meeting
Ann
doc5
server 2
doc1
doc2
Bob
server 3
doc3
Jane
doc4
server 4

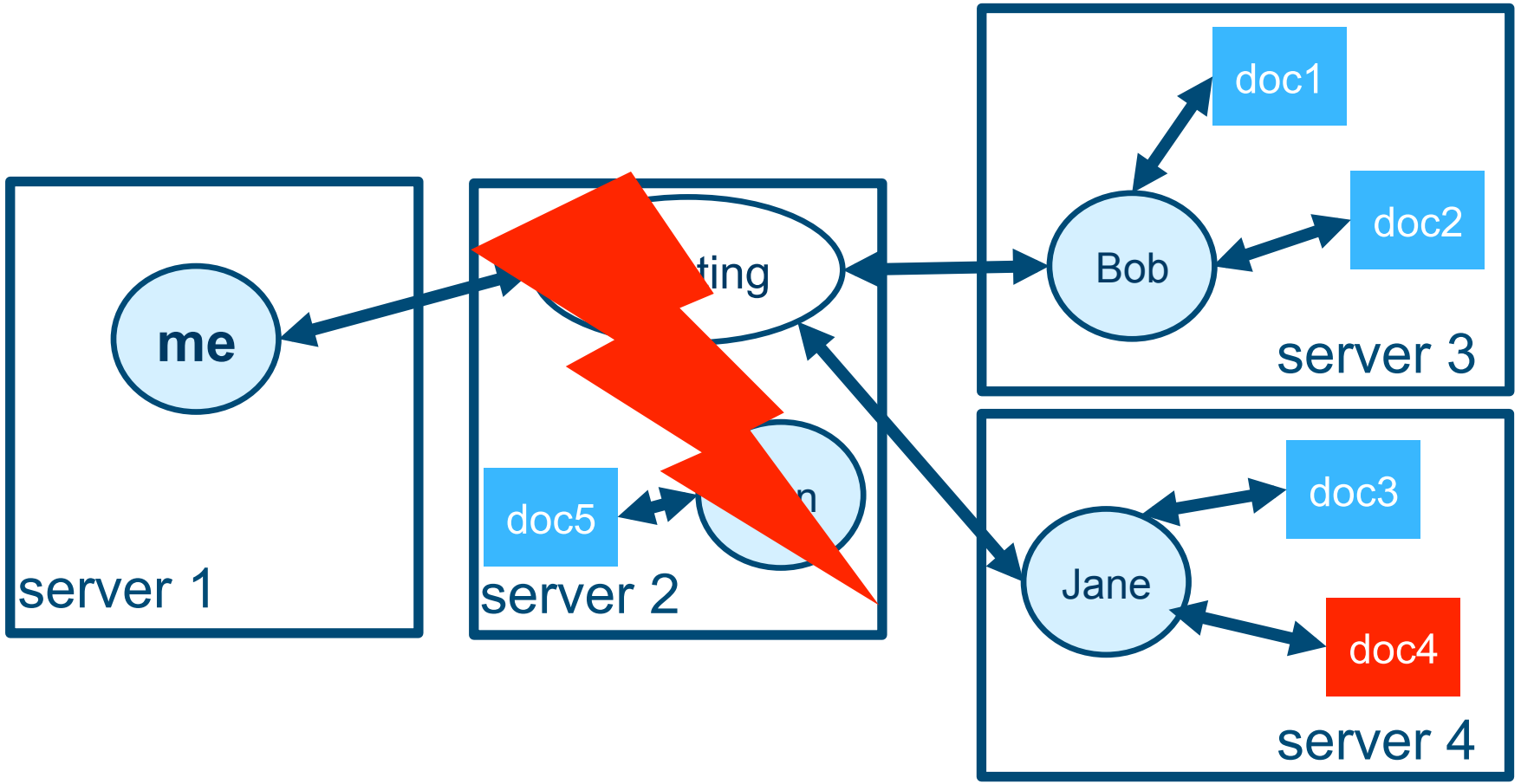
Latency


1
me
meeting
2
3
Bob
doc1
doc2
server 3
doc5
Ann
server 2
server 1
Jane
doc3
doc4
4
server 4

Freshness and consistency



Fault tolerance



It is hard to build low latency apps at scale

FARM simplifies building low latency apps

transactions and replication simplify programming

- hide failures, distribution, and concurrency
- abstraction that transactions run sequentially on a reliable server

no compromises: consistency, availability, performance

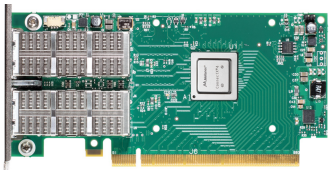
- strict serializability
- millions of transactions per second with sub millisecond latencies
- recovery from failure to peak performance in tens of milliseconds

FARM is enabled by three hardware trends

- inexpensive DRAM
  - currently $8/GB
  - machines with 128 GB, container will hold more than 100 TBs

- non-volatile RAM
  - NVDIMMs available today: DRAM+battery+SSD

- fast commodity networks with RDMA
  - CX3: 40 Gb/s and < 3 µs latency on Ethernet (CX4: 100 Gb/s)
  - CX3: 35M messages per second (CX4: 150M)



this hardware is commodity and being brought into datacenters

FARM software

- new algorithms and data structures optimized for RDMA
  - single-object read only transactions with a single RDMA read
  - Transaction commit protocol optimized for RDMA operations.
  - read validation during commit using a single RDMA read
  - btree and hash table lookups with a single RDMA read
- changes to OS and network card driver to speed up RDMA
  - efficient memory translation
  - connection multiplexing

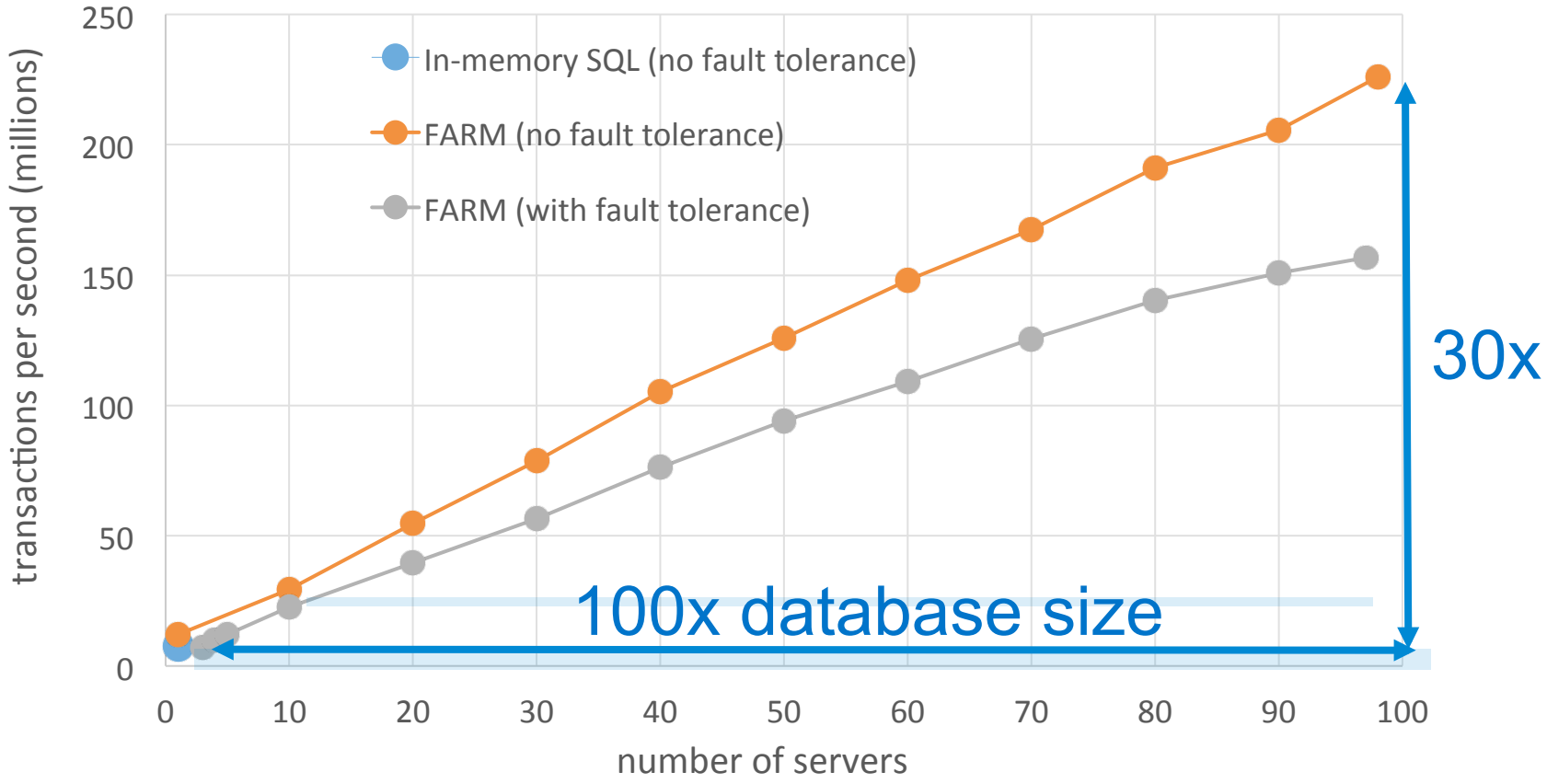
FARM API

Storage APIs

```
Transaction CreateTransaction()
ObjBuf *Transaction::Alloc(size, locality_hint)
ObjBuf *Transaction::Read(addr, size)
ObjBuf *Transaction::OpenForWrite(ObjBuf)
void Transaction::Free(ObjBuf)
void Transaction::Commit()
```

Communication APIs for application level coordination

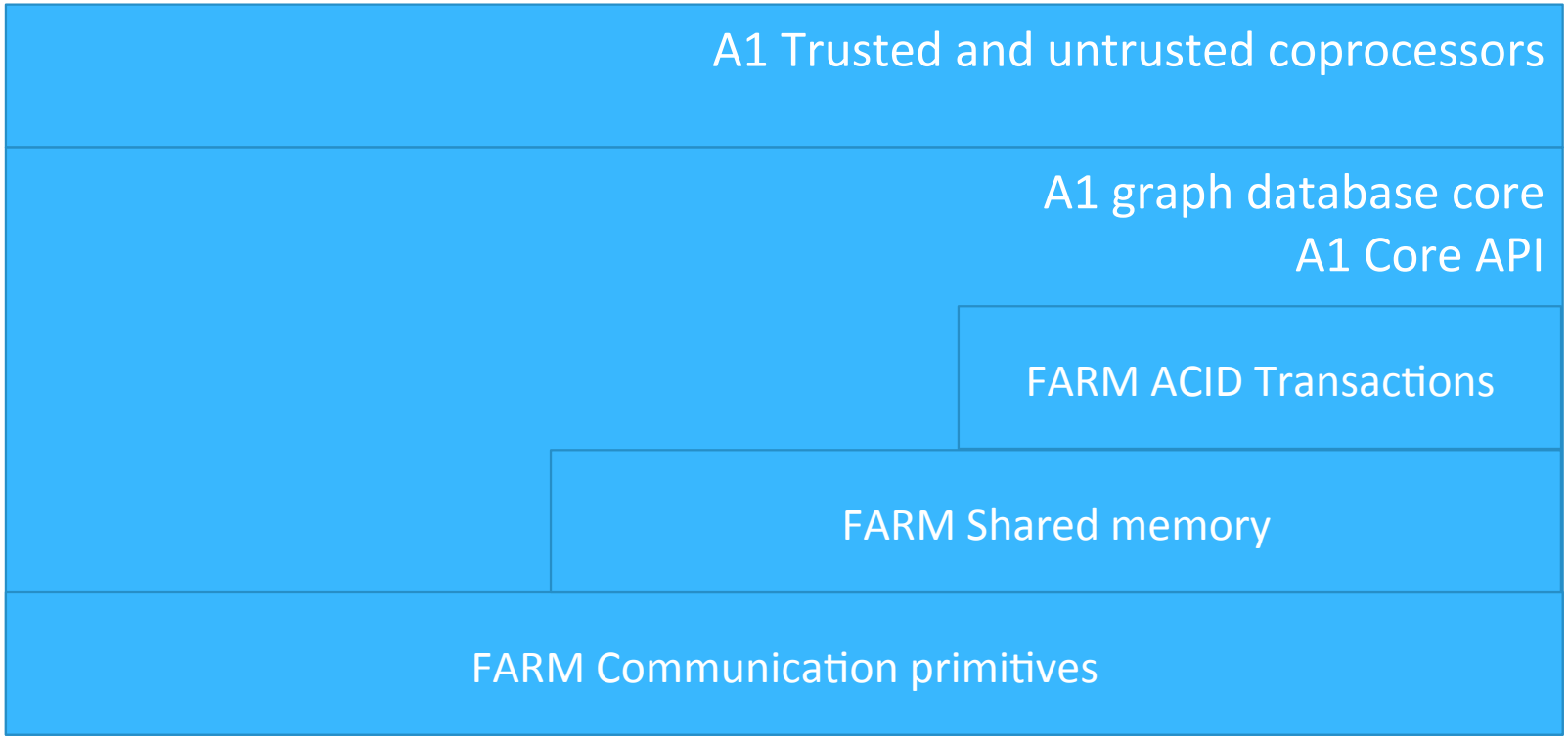
Scale-out OLTP throughput: TATP



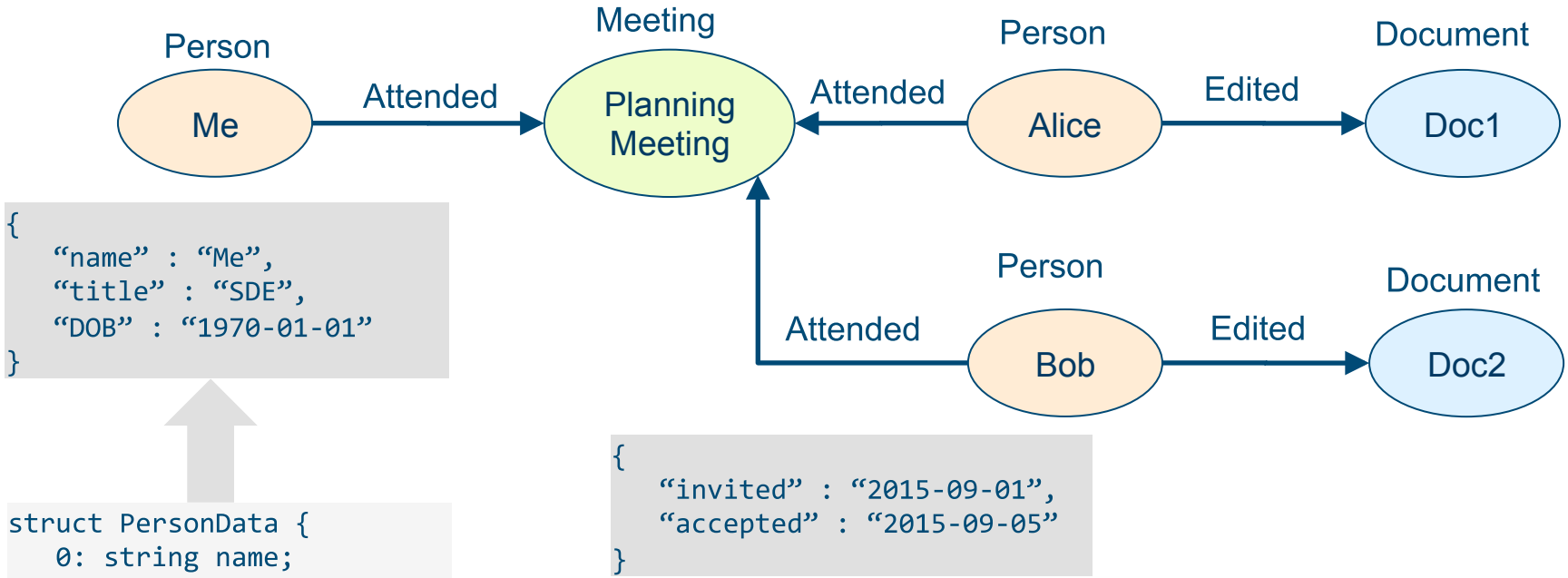
Why a Graph Database? On FARM?

- Many datasets naturally lend themselves to a graph structure
  - Facebook, Twitter, enterprise relationships
- Graph traversal queries are hard on relational stores
- A1 Goals
  - Scalable
  - Transactional
  - Complex traversal and subgraph queries
- Not targeted for graph compute

A1 Architecture overview



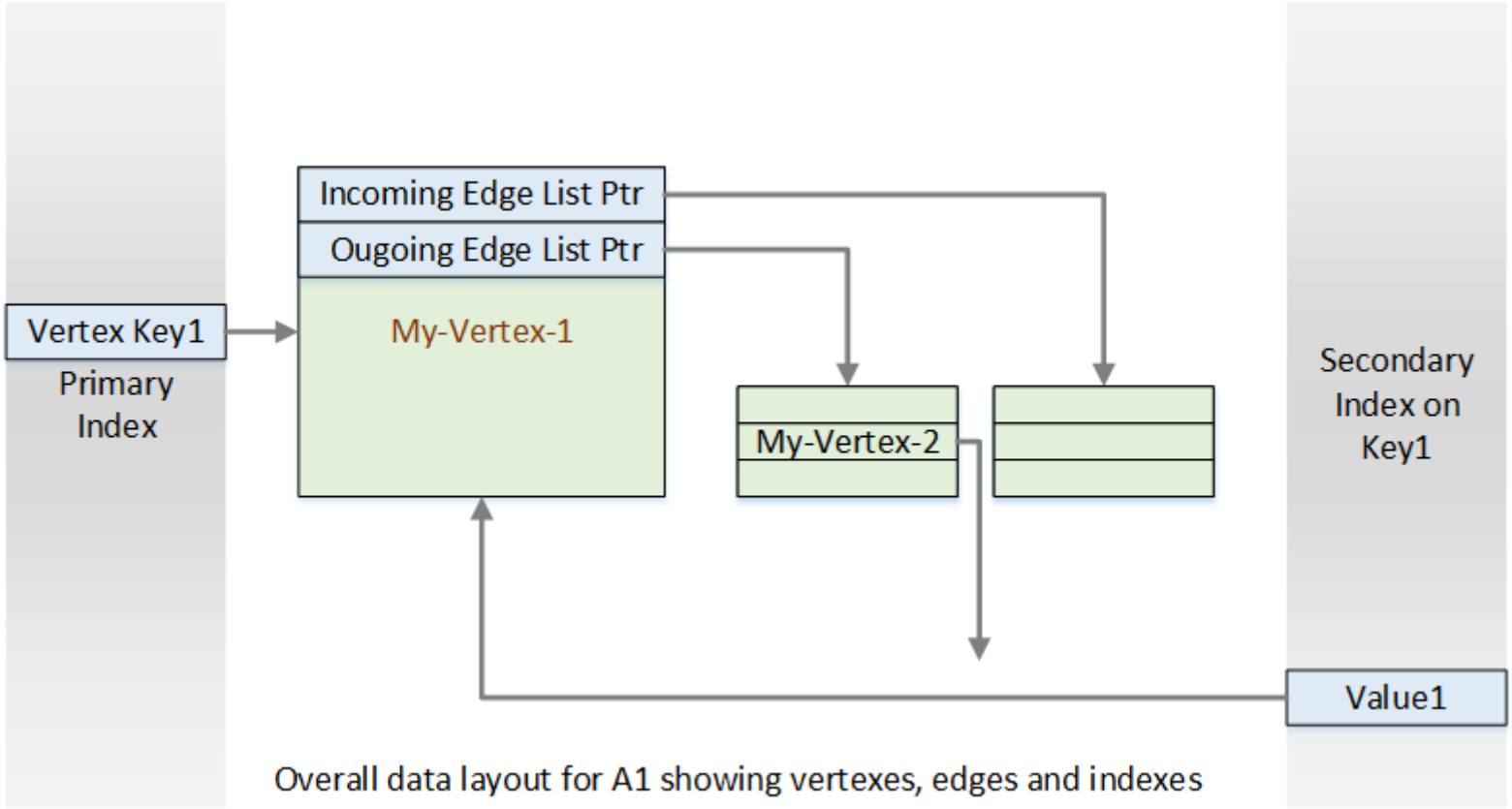
The A1 Data Model



A1 Data Layout

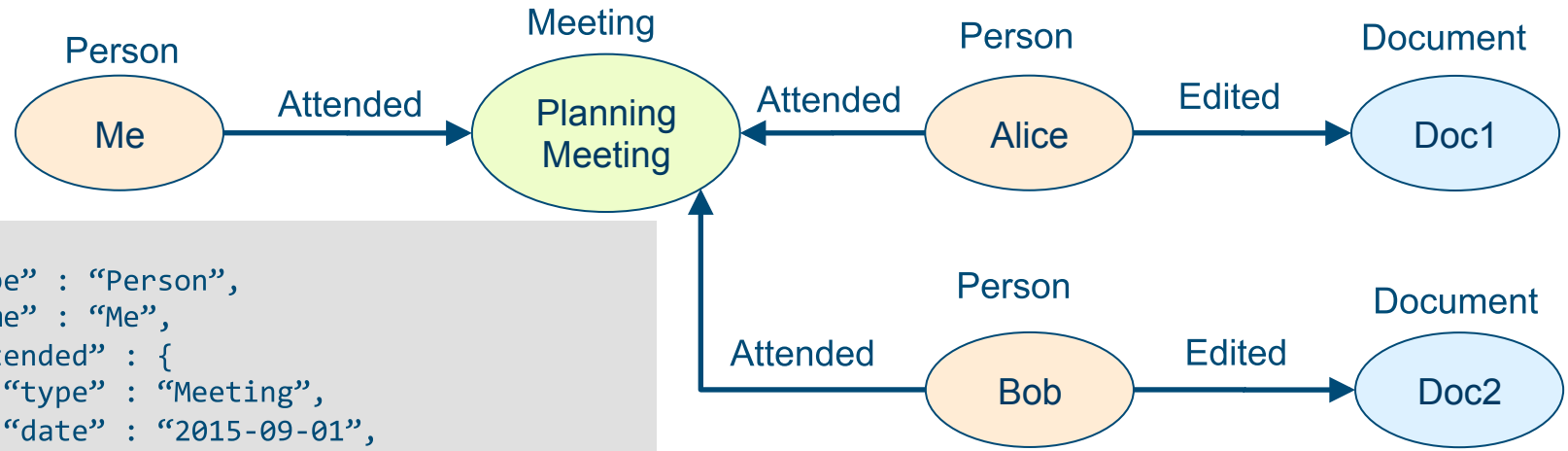
- Locality of reference matters
- Graph : a set of vertexes with adjacency lists
- Building blocks
  - FARM objects identified by 64 bit pointers
  - RPC primitives
  - BTree for indexes
  - Linked lists for adjacency

A1 Data Layout



Overall data layout for A1 showing vertexes, edges and indexes

BFS Query




```
{
   “type” : “Person”,
   “name” : “Me”,
   “Attended” : {
       “type” : “Meeting”,
       “date” : “2015-09-01”,
       “Attended” : {
           “type” : “Person”,
           “Edited” : {
               “type” : “Document”
           }
       }
   }
}
```

Traversal Algorithm

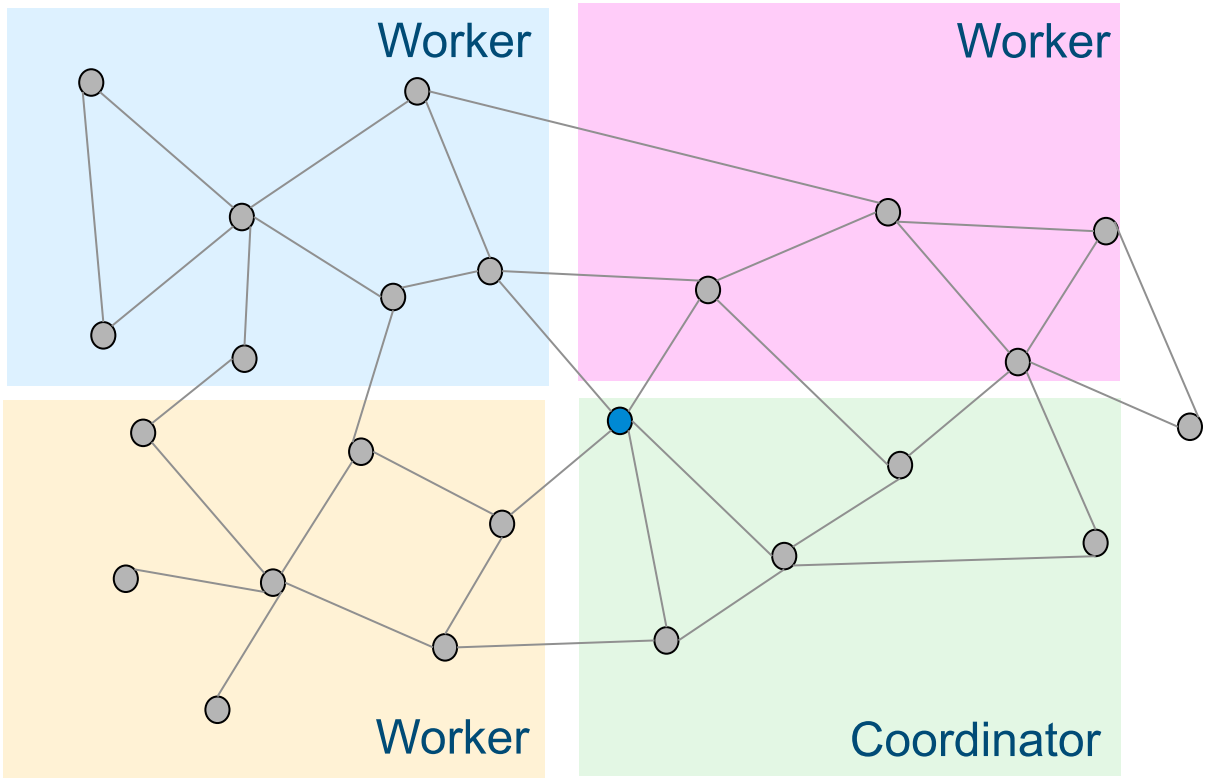
- Designate starting host as coordinator (maintain visited list)
- Coordinate level by level traversal

  - Data Shipping

  - Query shipping
    - RPC for synchronization
    - All predicates applied locally
    - All neighbors discovered locally

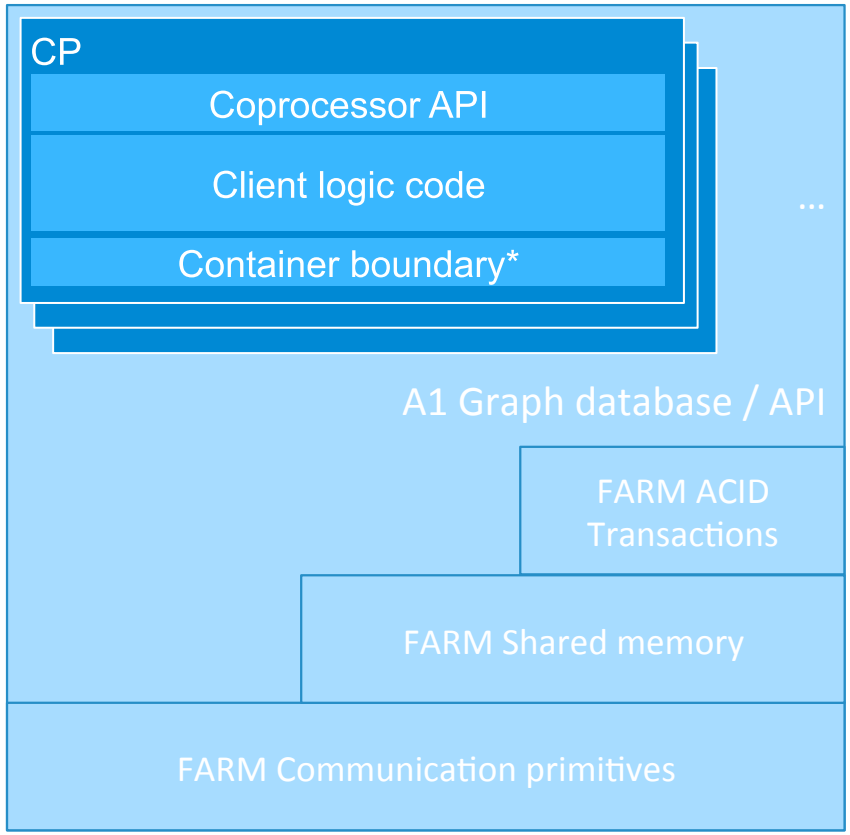
  - Hybrid

BFS Traversal via Query Shipping



1. Start
2. Find neighbors
3. Coord: Partition and ship query
4. Worker: Find neighbors
5. Worker: Ship list back to coordinator
6. Back to step 3

Extensibility by Coprocessors



Coprocessor models

Trusted – core extensions

No separation, running inside the FARM process space
No latency

Untrusted core hosted

Separate address and process space on same machine.
1-2us (10-25% overhead)

Untrusted cluster hosted

Separate machines on the same failure/network domains
50-200 us (4x – 16x overhead)

Preliminary Performance

Linkbench with 1B vertexes and 4.5B edges

- 3X replication
- Single container / network domain / failure domain

Single transactions

Vertex Creation: 375 us
Edge Create: 495 us
Edge Get: 240 us
Vertex Read: 100 us

BFS

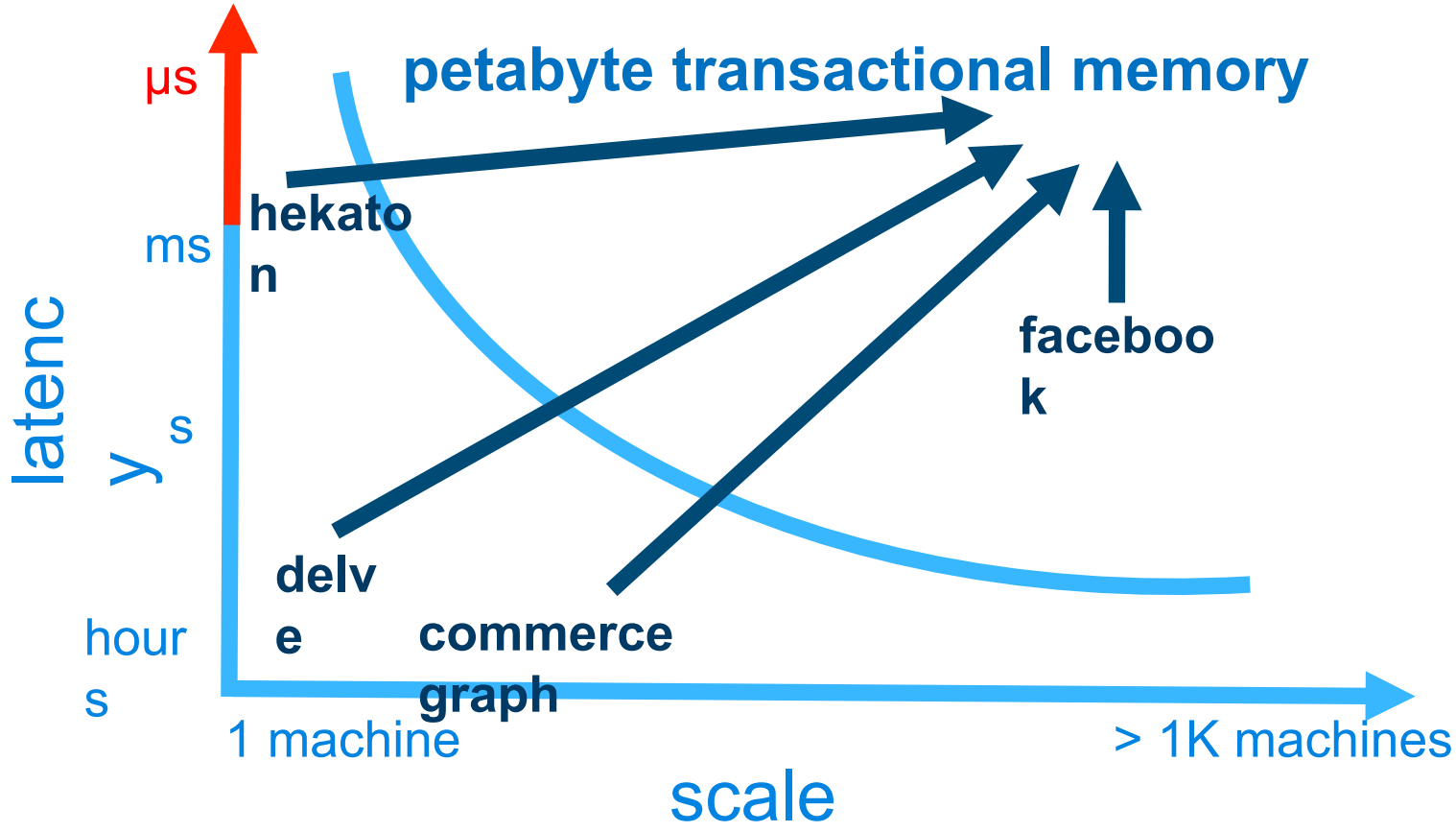
1 hop : 700 us
2 hop : 850 us

Throughput

~ 350K transactions/second/ node
Linear scale-up to 100 nodes

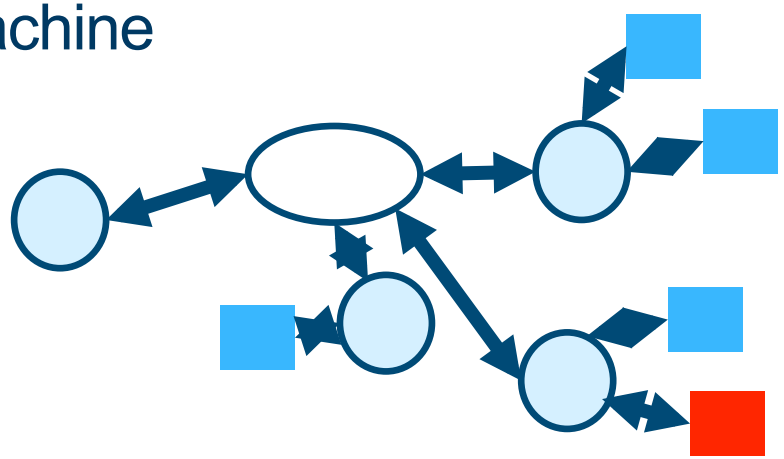
Backup slides

Important applications can benefit



Petabyte transactional memory makes it easy

- general transactional memory model
  - objects, arrays, and pointers
  - graphs, relational databases, key-value stores on the same system
- abstraction of running on a single machine
  - scale out
  - low latency
  - freshness and consistency
  - fault tolerance
- high throughput to keep costs low

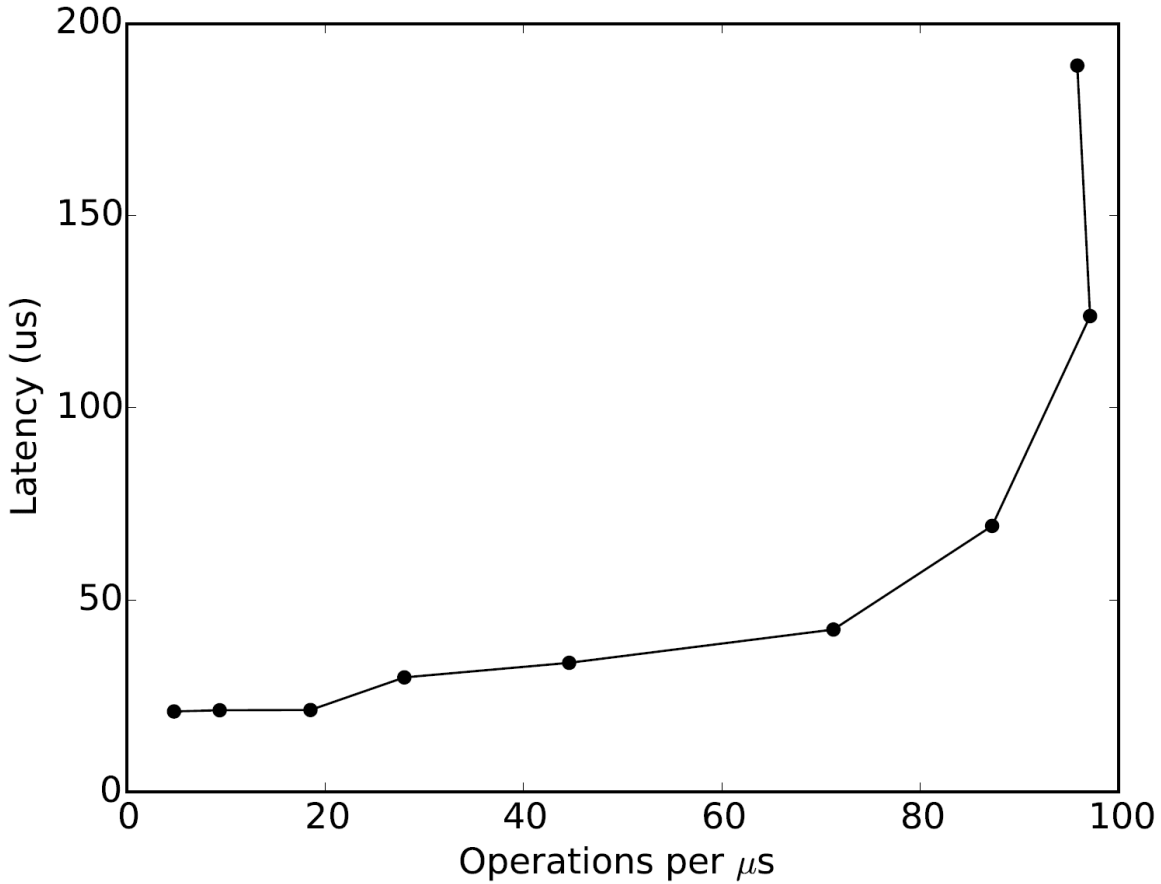


it will speed up innovation as MapReduce did for batch analytics

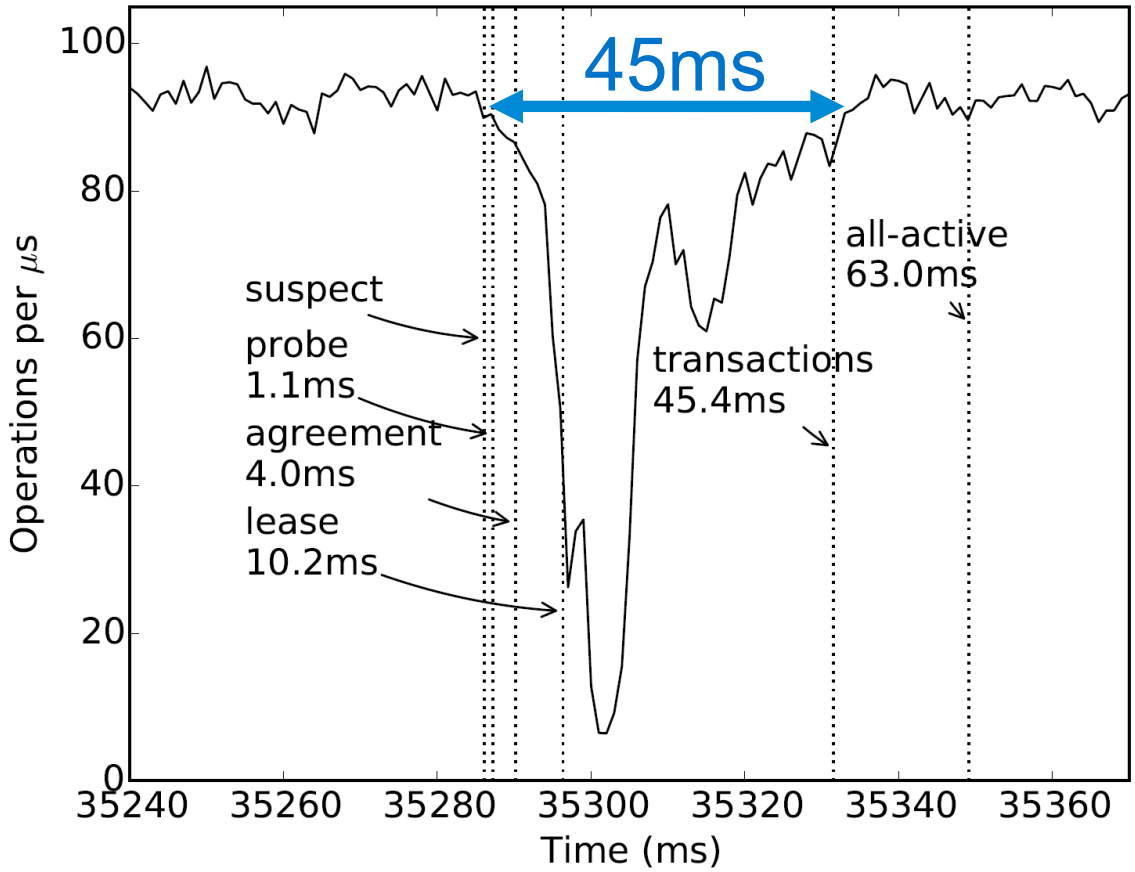
Data Model Comparison

Data Model	Horizontal Scaling	Join Support	Language Support	Maturity
Key-Value	Easy	None	None	Medium
Relational	Non-trivial	Good	SQL	High
Graph	Hard	Join-less	Specialized	Low

Scale-out OLTP throughput-latency: TATP



Scale-out OLTP recovery: TATP



Scale-out OLTP: TPC-C benchmark