Paxos Replicated State Machines as the Basis of a High-Performance Data Store

William J. Bolosky,
Dexter Bradshaw, Randolph B. Haagens,
Norbert P. Kusters and Peng Li

March 30, 2011

Q: How to build a fault-tolerant, high-performance data store from commodity parts?

A: Paxos replicated state machines

- Paxos Replicated State Machines
  - Sequentially consistent
  - Persistent
  - Fault tolerant
  - Don’t rely on clock sync for correctness
  - Thought to be too slow
- Conventional systems compromise on
  - Semantics (*e.g.* data consistency after failures)
  - Assumptions (*e.g.* clock sync for correctness)
  - API (*e.g.* append only)
  - Special hardware (*e.g.* FAB’s write timestamps)
- Paxos equaling the speed of a conventional system is a win
  - That we sometimes do better is a bonus

Take Away Point

- For datacenter-like systems that:
  - Value **C**onsistency and **A**vailability over **P**artition tolerance
  - Have operation latencies ≥ network latencies
- Paxos replicated state machines
  - Perform very well
  - While not compromising

Outline

- **Background: Replicated State Machines and Paxos**
- SMARTER and Gaios
- A new protocol for read-only operations
- Performance evaluation and comparison to primary-backup replication

Replicated State Machines

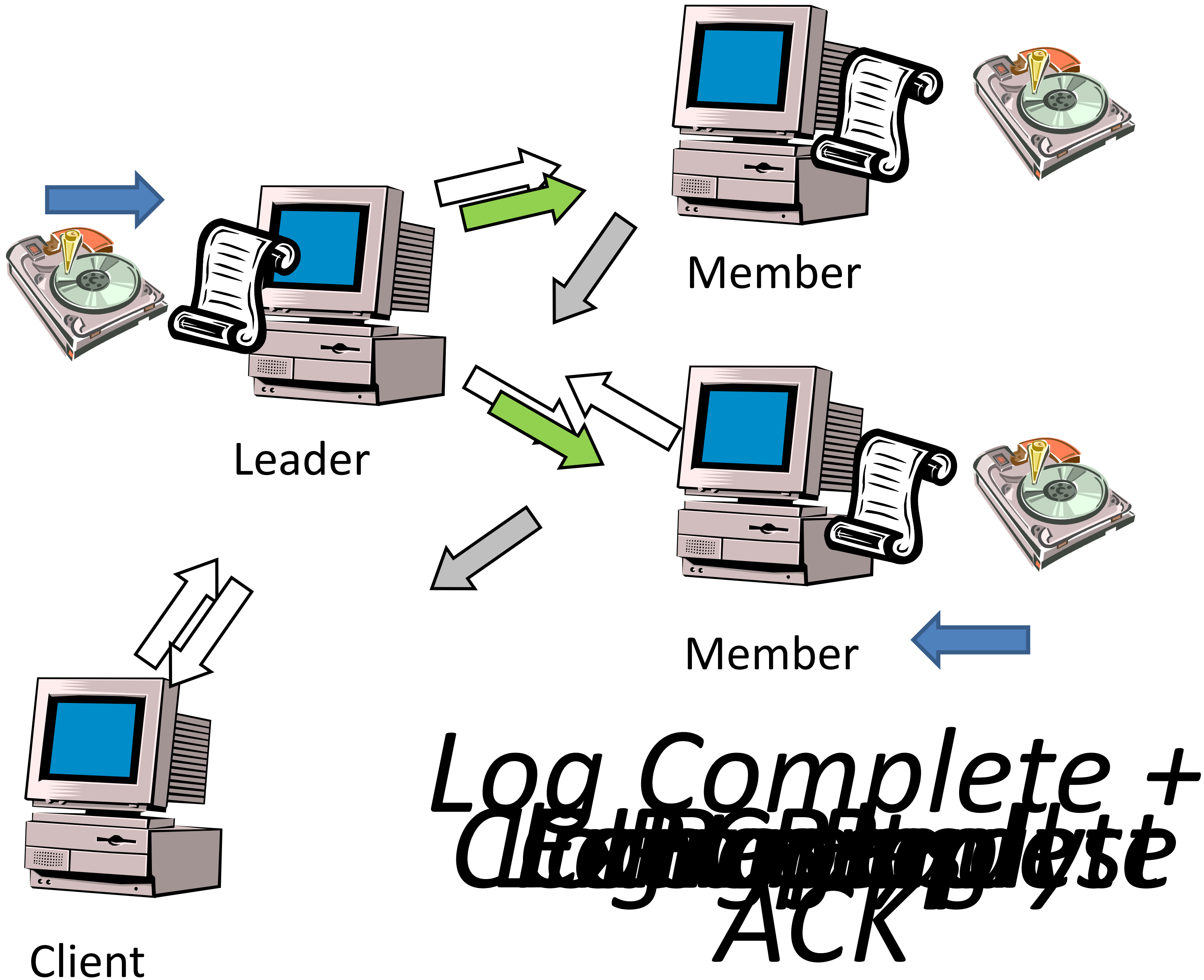
- For fault tolerance
  - Of any deterministic computation
  - Via replication
  - Replicas see the same sequence of inputs
- Paxos is a protocol for guaranteeing input ordering, even with:
  - Multiple clients
  - Unreliable networks
  - No synchronized clocks
  - Unlimited machine reboots
  - Some permanent stopping faults (*i.e.*, disk losses)
  - But not Byzantine faults

Non Trade-Off

- RSMs’ one-at-a-time execution model seems to be at odds with disks’ need to reorder IO for efficiency. It’s not.
- Analogous to an out-of-order processor.

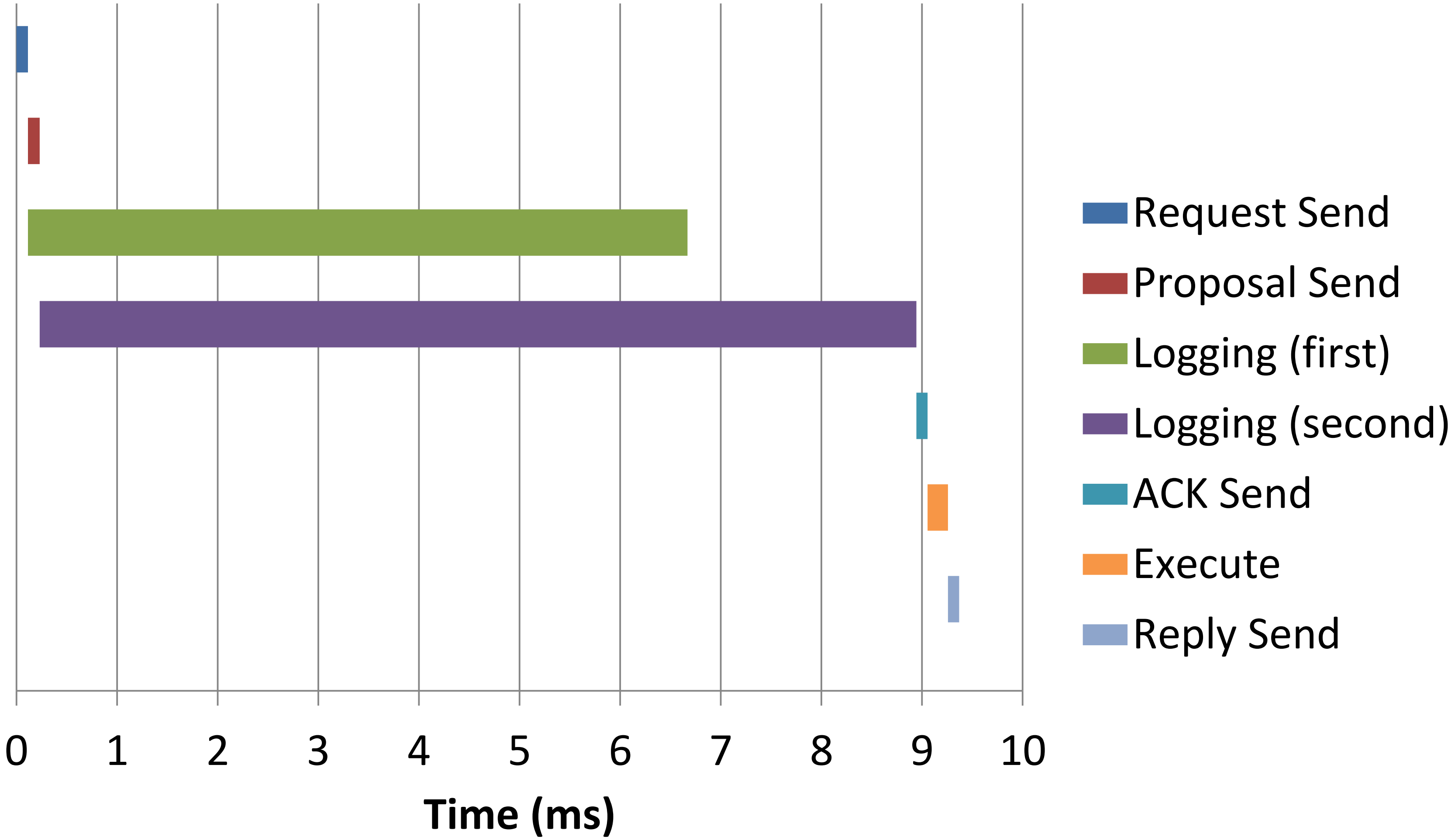
Paxos Basics

- Paxos binds client requests to sequentially numbered *slots*.
- In normal operation requires a write to persistent store to survive power loss.
- Has a dynamically selected and changeable *leader* that drives the protocol.


Member
Leader
Member
Client
Log Complete +
ACK

4K Write Latency Timeline

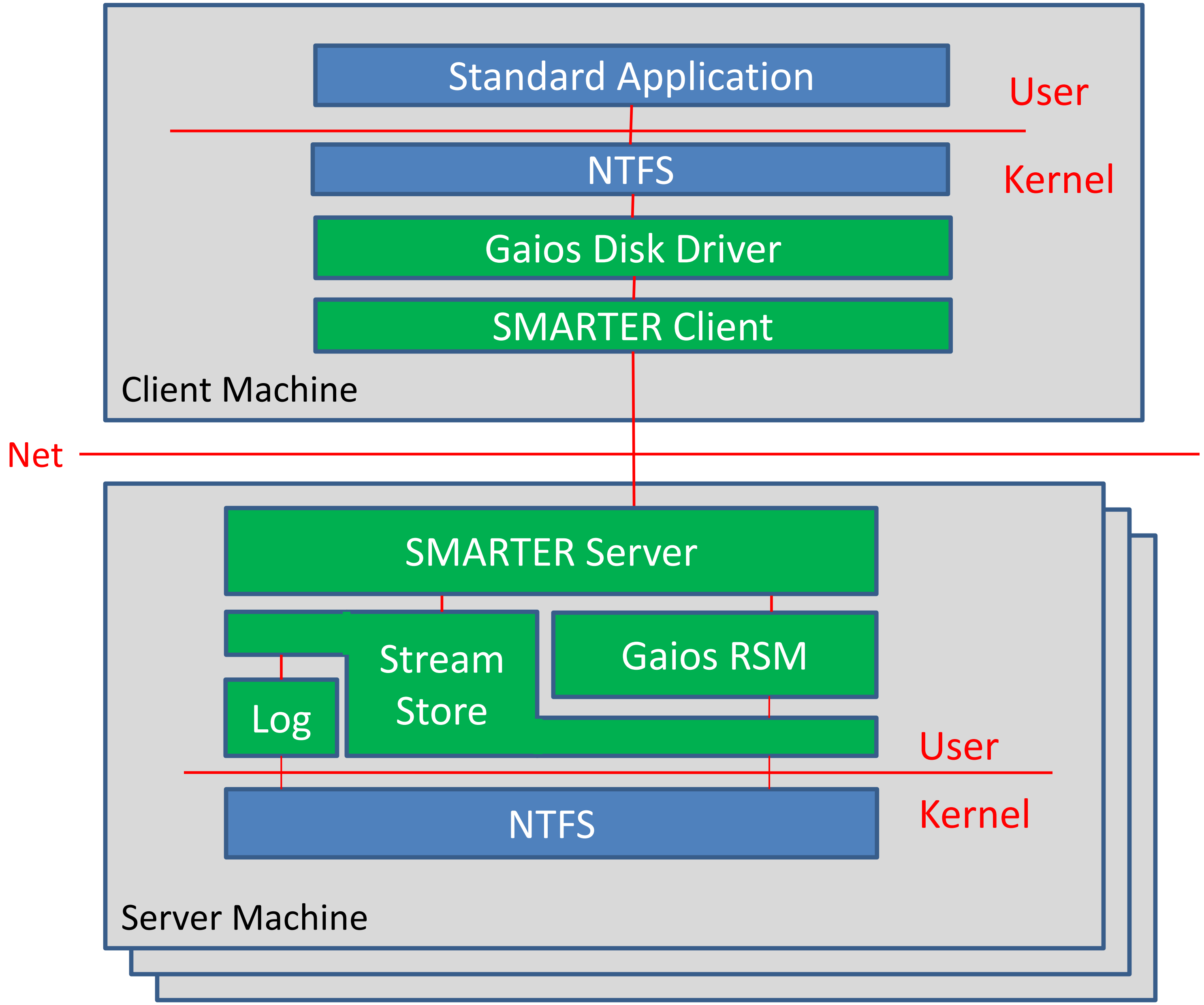
(One-at-a-Time Operations)



Outline

- Background: Replicated State Machines and Paxos
- **SMARTER and Gaios**
- A new protocol for read-only operations
- Performance evaluation and comparison to primary-backup replication

Gaios Architecture



Getting Efficiency

- Mostly just lots of good engineering
  1. Pipelining
  2. Batched write behind
  3. Overlap fetching with logging
  4. Batching client requests
  5. Zero-copy data path
- Novel read-only operation protocol that allows consistent reads from any node

Outline

- Background: Replicated State Machines and Paxos
- SMARTER and Gaios
- **A new protocol for read-only operations**
- Performance evaluation and comparison to primary-backup replication

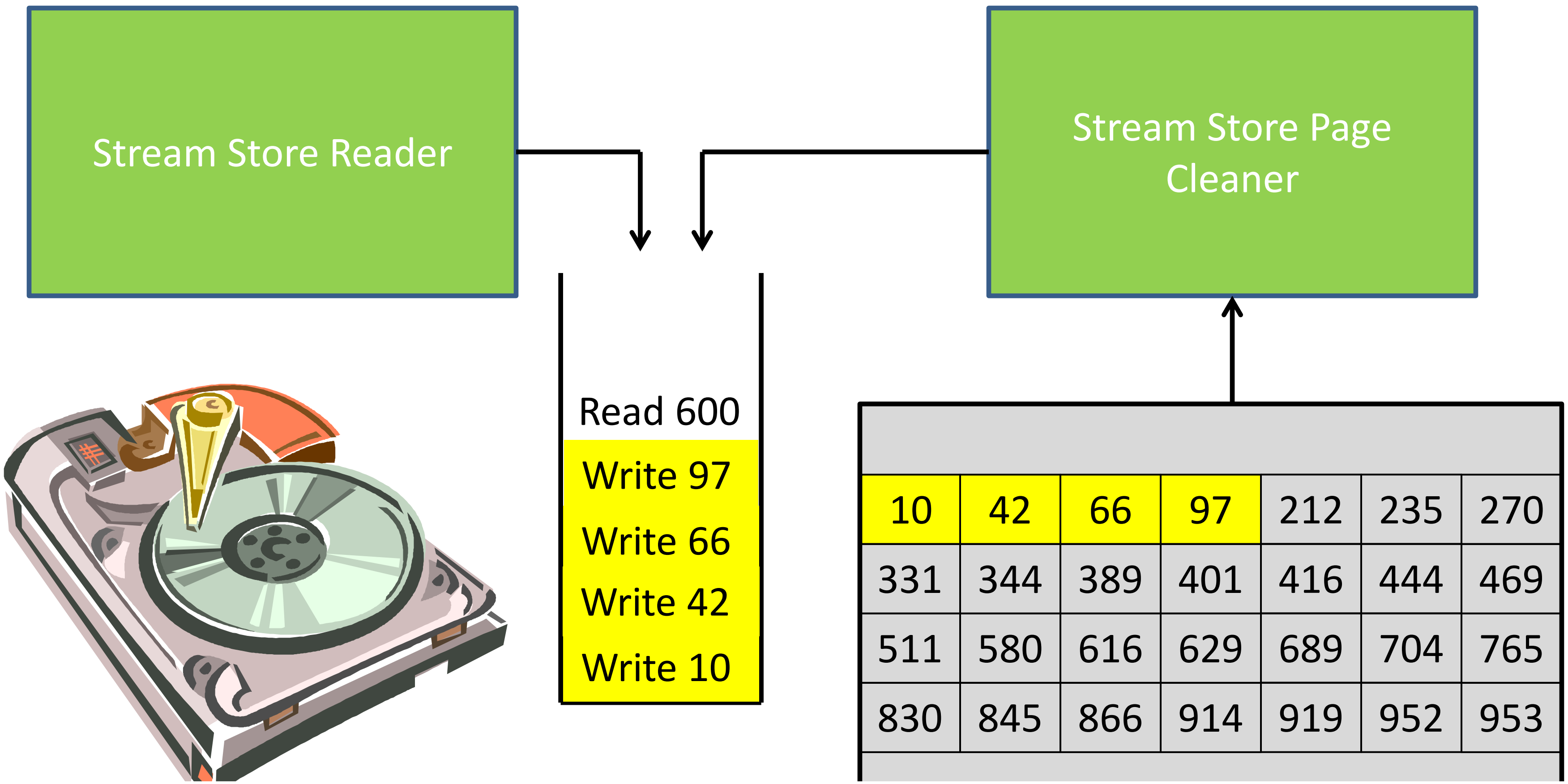
Read Consistency Property

Not-Before Constraint: *When a read-only request **R** completes, it reflects any data known by any client to be written at the time **R** was sent.*

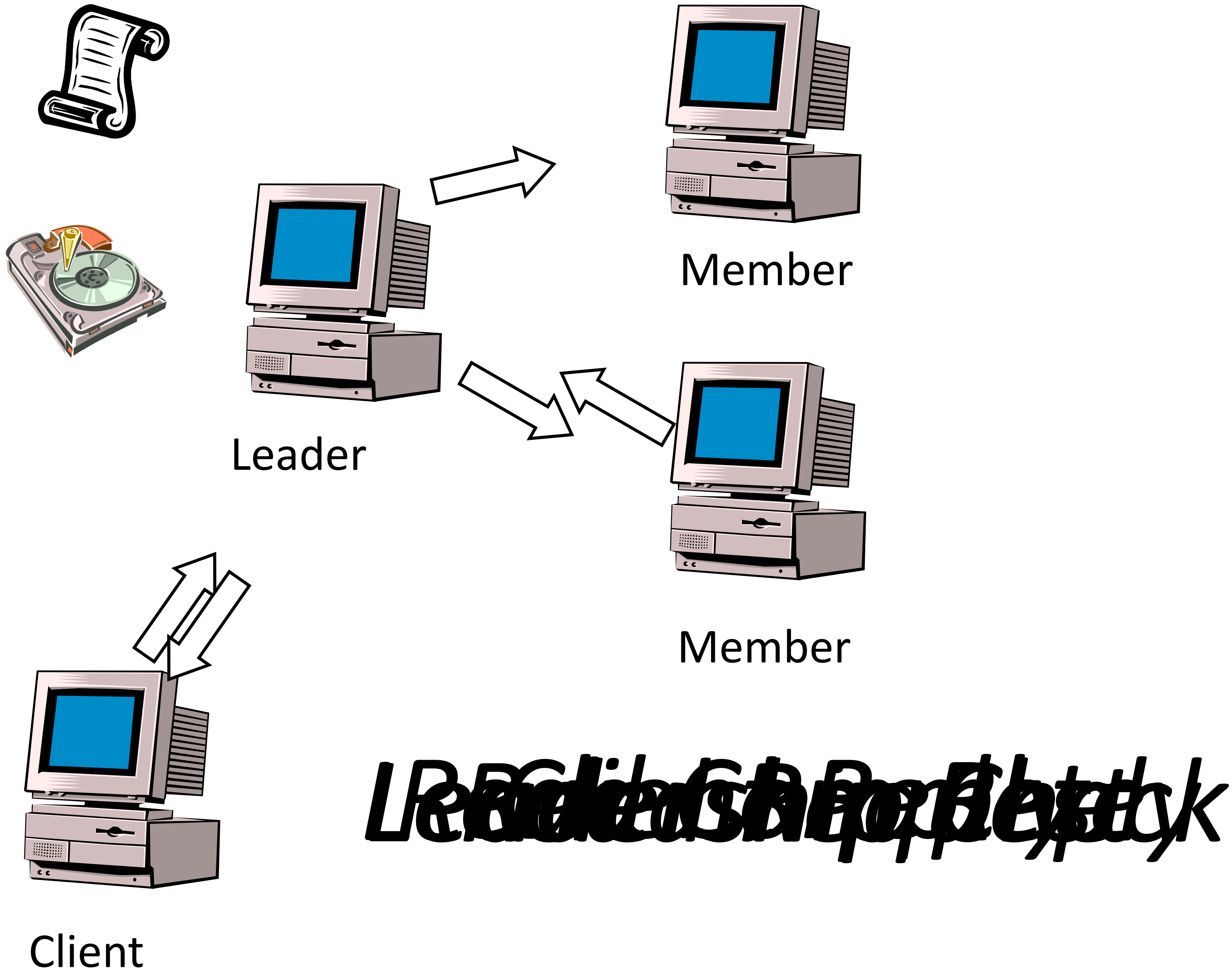
Read-Only Operations

- Read-only operations only need to run in one place
- Using all disks is crucial
- Dynamically selecting location helps
  - Avoid nodes that are writing

Read/Write Contention



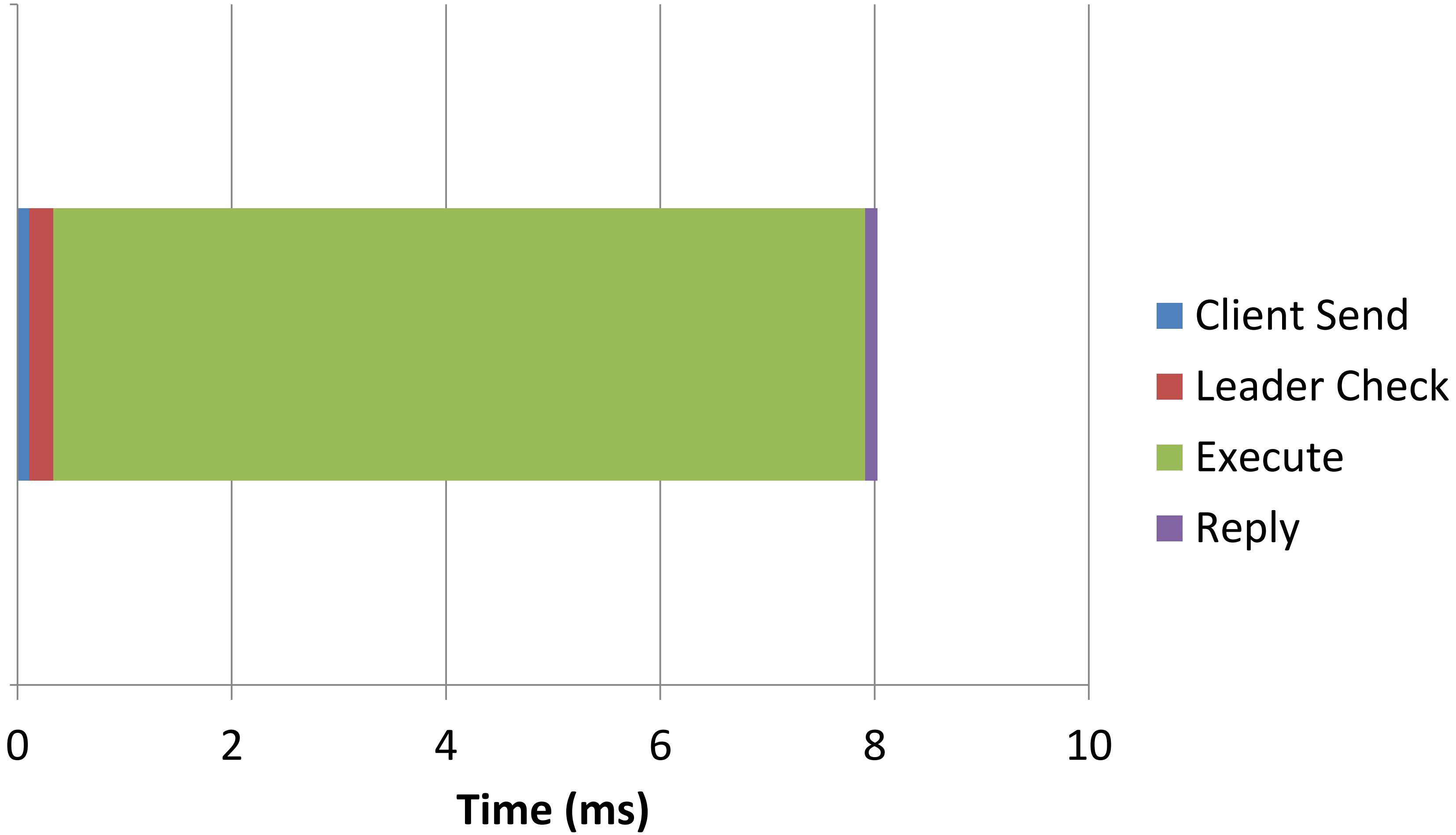
Randomize Checkpoint timing across nodes



Member
Leader
Member
Client

4K Read Latency Timeline

(One-at-a-Time Operations)



Outline

- Background: Replicated State Machines and Paxos
- SMARTER and Gaios
- A new protocol for read-only operations
- **Performance evaluation and comparison to primary-backup replication**

Primary-Backup Replication

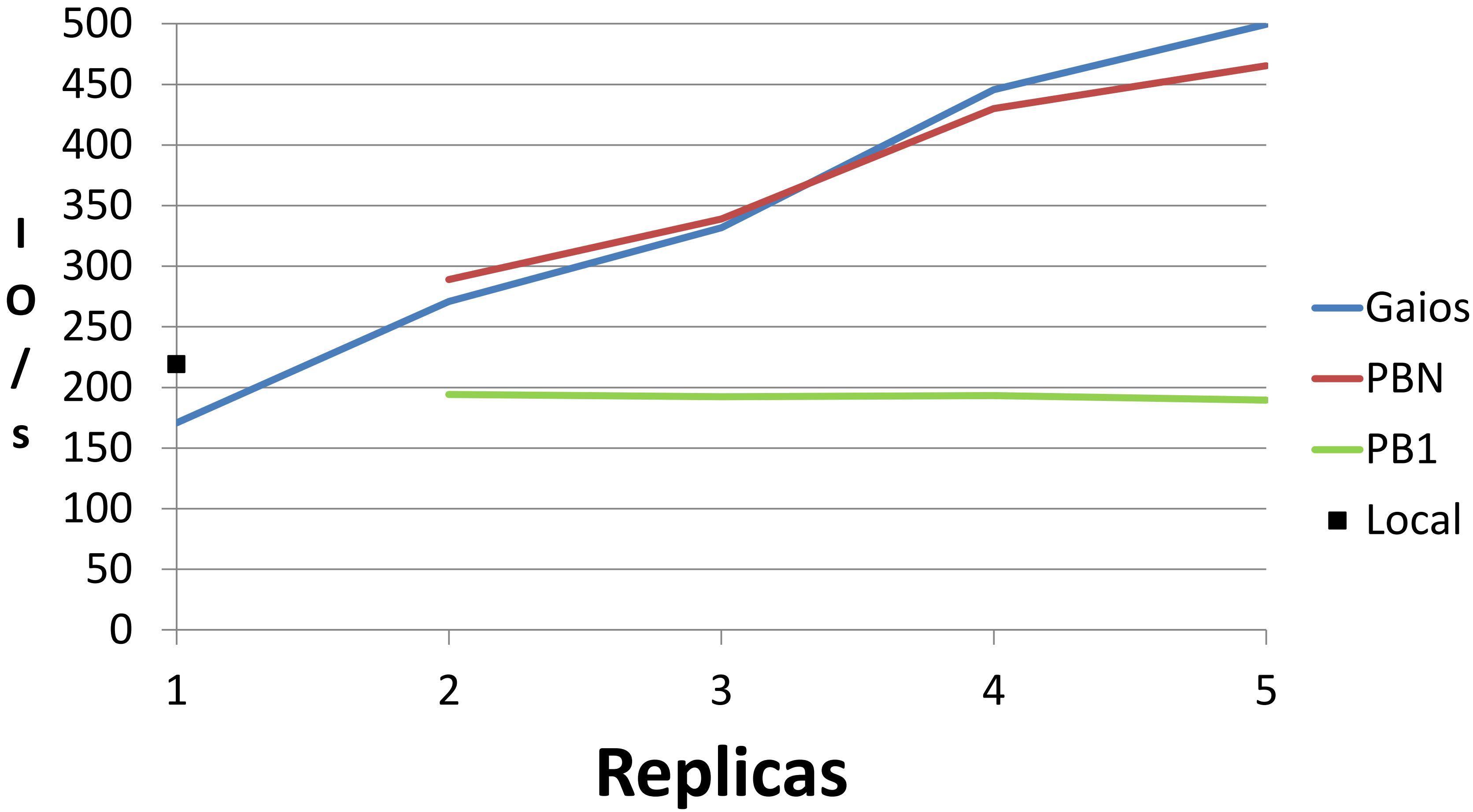
- (Usually) Sends both read and write replies from the primary in order to achieve the read consistency property
- Uses leasing protocol for primary
  - No need for a quorum check on reads
  - Relies on clock sync for correctness, which in practice means it trades failover time for correctness

Read Distribution

- Primary-Backup forces reads to one node, while SMARTER spreads them across all, which can matter for random reads
- P-B can achieve spreading by striping data across many groups and locating the primaries on different nodes; this spreading is static
- Implemented two versions of P-B:
  - Worst-case PB1 where all reads come from one node
  - Best-case PBN which uses round-robin reads

8K Random Read Throughput

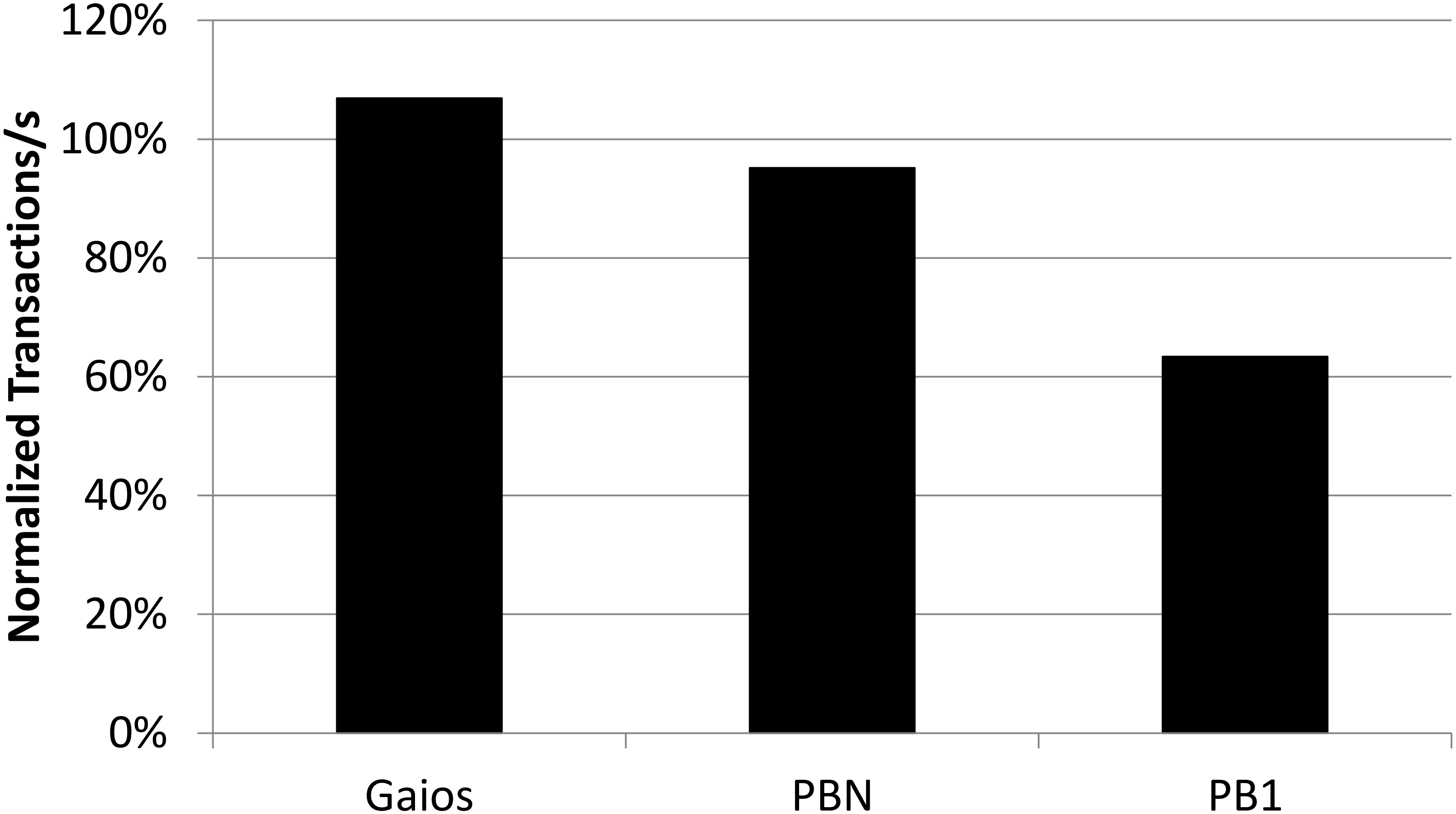
(Lots of outstanding operations)



Transaction Processing

- Ran industry standard OLTP load over Microsoft SQL Server 2008.
- Critical factors: SQL log write latency, random read bandwidth.
- Even read/write ratio, mostly ~8K.

OLTP Performance
(3 nodes, 50% read workload)



Conclusion

- Paxos RSMs are fine for high-performance disk-based applications, it just takes careful engineering.
- In some cases, they outperform best-case P-B due to flexibility in directing reads.
- There is no need to compromise on semantics, buy special hardware, depend on clocks, *etc*.

Photo of Gaios, Paxos, Greece