Consistent Replication in Distributed Multi-Tier Architectures

Thomas Repantis
Akamai Technologies

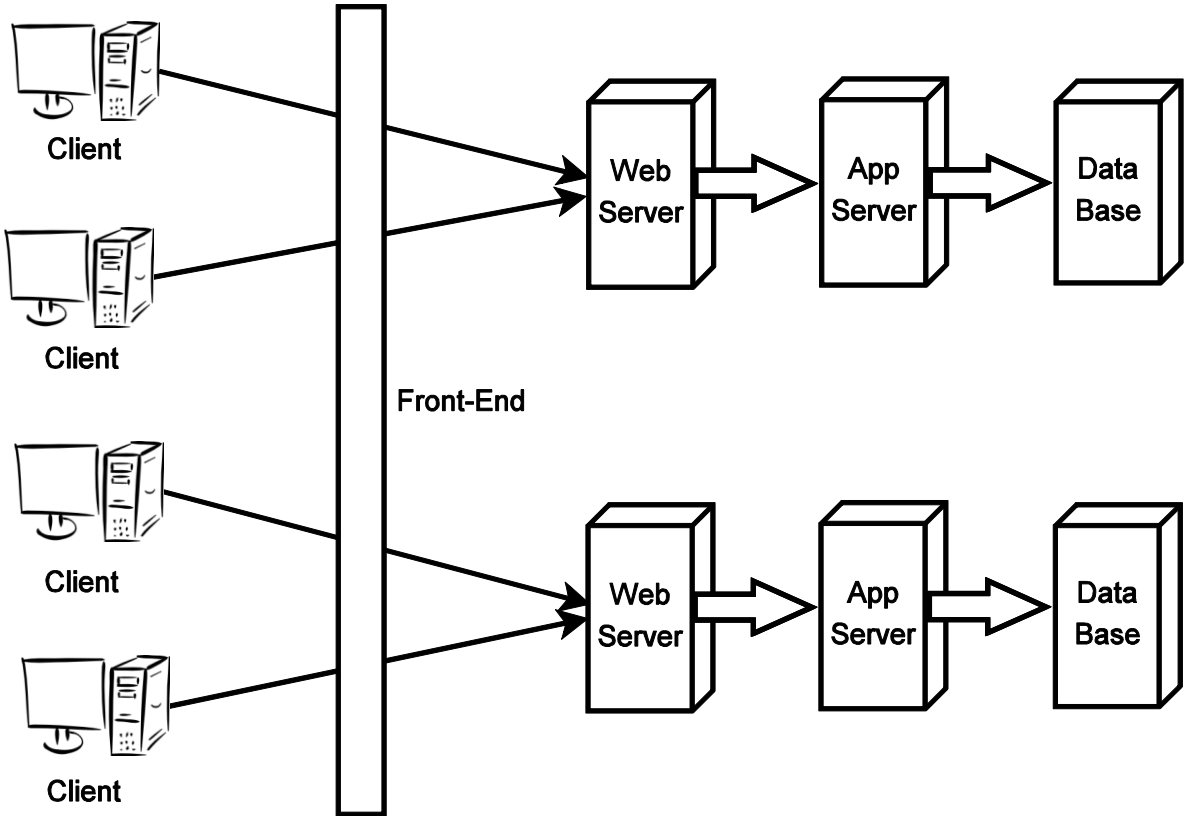
Arun Iyengar
IBM Research

Vana Kalogeraki
AUEB Greece

Isabelle Rouvellou
IBM Research

trepanti@akamai.com

Distributed Multi-Tier Architectures



- E-commerce
- E-banking
- Social networking
- Blogs
- Wikis

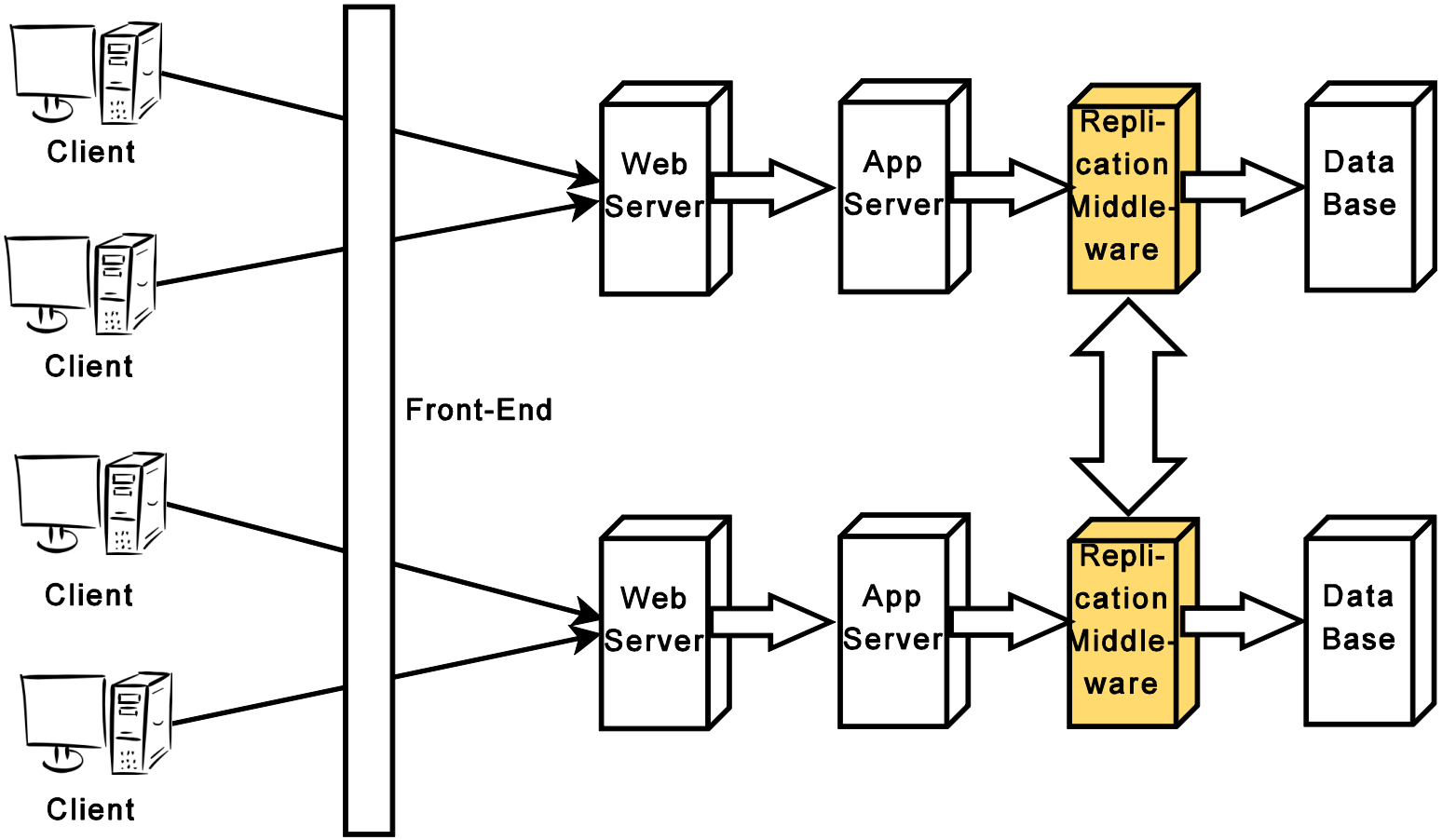
Replication and Consistency

- Replication increases scalability and availability
- Problem: Consistently maintain replicated data
- Strong consistency
  - All copies of a data object served at any time are identical
  - E.g., e-banking, online retail stores, auction marts
- Weak consistency
  - Different nodes can serve different versions of the same data object at the same time
  - E.g., online forums, social networking

Contributions

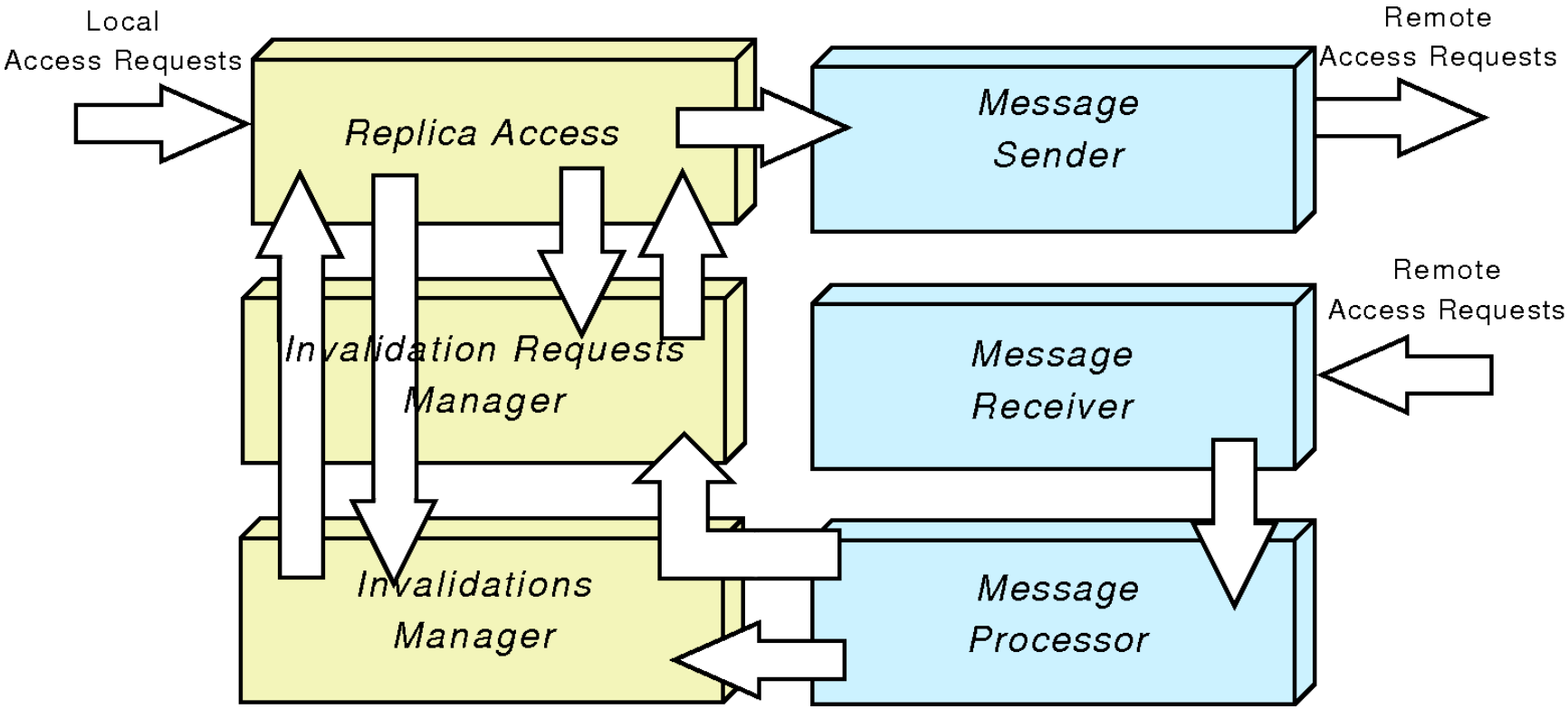
- Propose an efficient, distributed, strong consistency protocol
- Implement a Java multi-threaded replication middleware
- Performance study of common replication approaches using the TPC-W benchmark
  - No replication
  - Partitioning
  - Weak consistency
  - Lock-based strong consistency
  - Our invalidation-based strong consistency

System Model

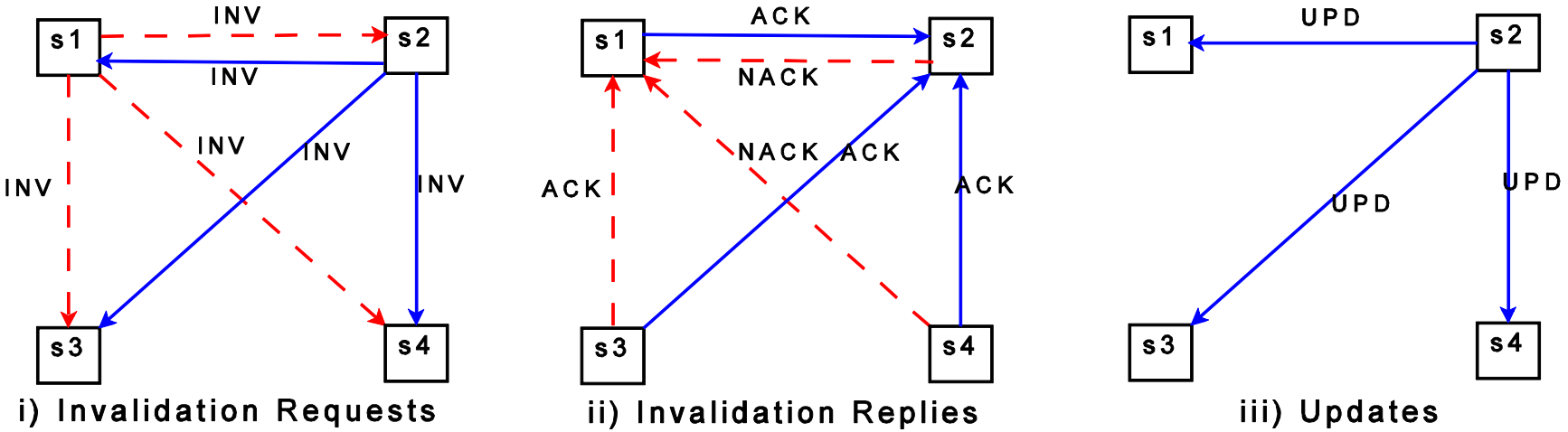


Active or passive interception

Replication Middleware



Invalidation-Based Strong Consistency



- Only after collecting positive replies from all nodes an update can proceed
- A request with earlier timestamp is rejected and gets negative reply

Submit Update Request (Sender)

Input:
- Local invalidation request INVt for table t
- Number of replicas S

Output:
- False (cancel request) or
- True (grant request and send update UPDt)

```
for each replica si in S
    send invalidation request INVt to si
invalidation replies = 0
while invalidation replies < S
    receive invalidation reply
    invalidation replies++
    if invalidation reply is negative (NACKt)
            return False
return True
```

Approve Update Request (Receiver)

Input:

- Remote invalidation request INVt for table t

Output:

- Positive invalidation reply ACKt or
- Negative invalidation reply NACKt

if (exists local or remote invalidation request INVt'
and timestamp(INVt') > timestamp(INVt))
  return NACKt
else
  return ACKt

Other Common Consistency Protocols

- No replication
- Weak consistency
- Partitioning
- Lock-Based strong consistency

Weak Consistency

- Different nodes can serve different versions of the same data at the same time
- Concurrent reads and concurrent writes allowed
- Nodes continue serving while receiving and applying updates
- Good performance, never block
- No accurate global state

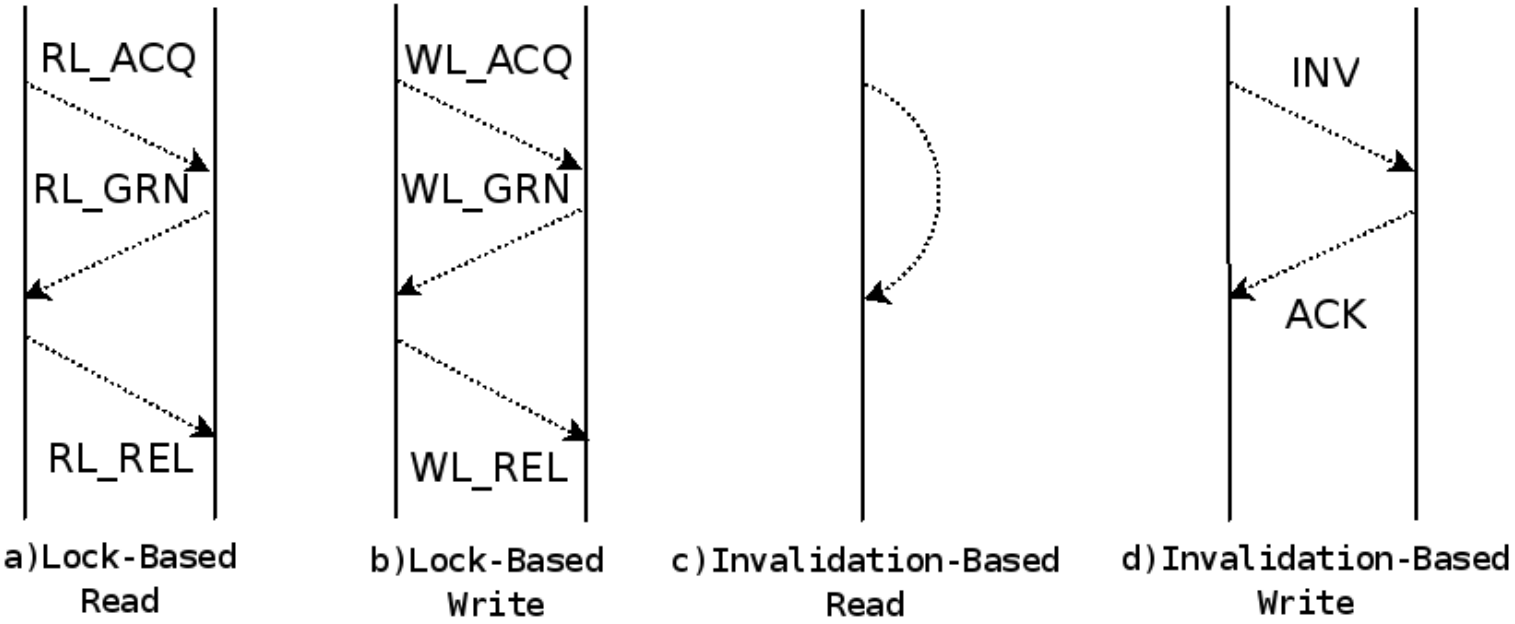
Partitioning

- Partition data among nodes
  - Different tables on different nodes
  - Distribute a table's rows across nodes
  - Distribute a table's columns across nodes
- Good performance, never block
- Strong consistency, but no replication
  - Scalability limits
  - Availability may suffer
- Queries may touch multiple nodes

Lock-Based Strong Consistency

- Readlocks and writelocks
  - Concurrent reads are allowed
  - No concurrent writes or reads and writes
- Centralized implementation
  - Single lock manager
  - Single point of failure and potential bottleneck
- Distributed implementation
  - Acknowledgements from all nodes with copies
  - Requires more messages

Lock-Based vs Invalidation-Based

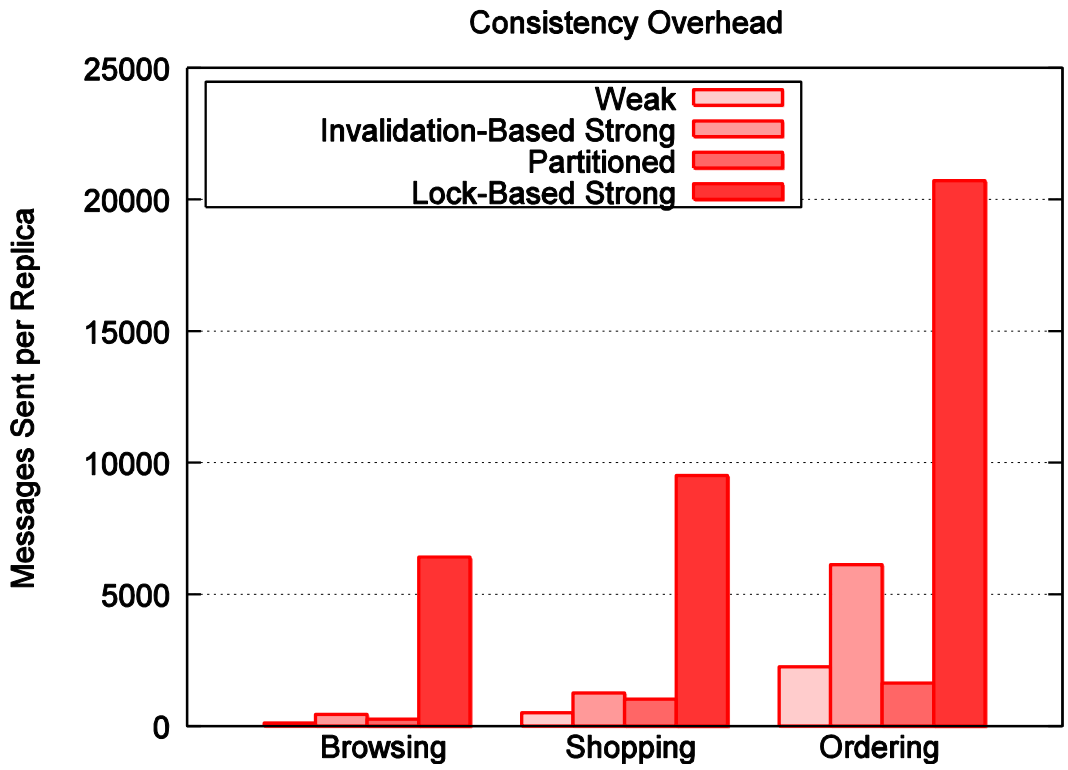


Invalidation-based requires no synchronization for reads

Experimental Setup

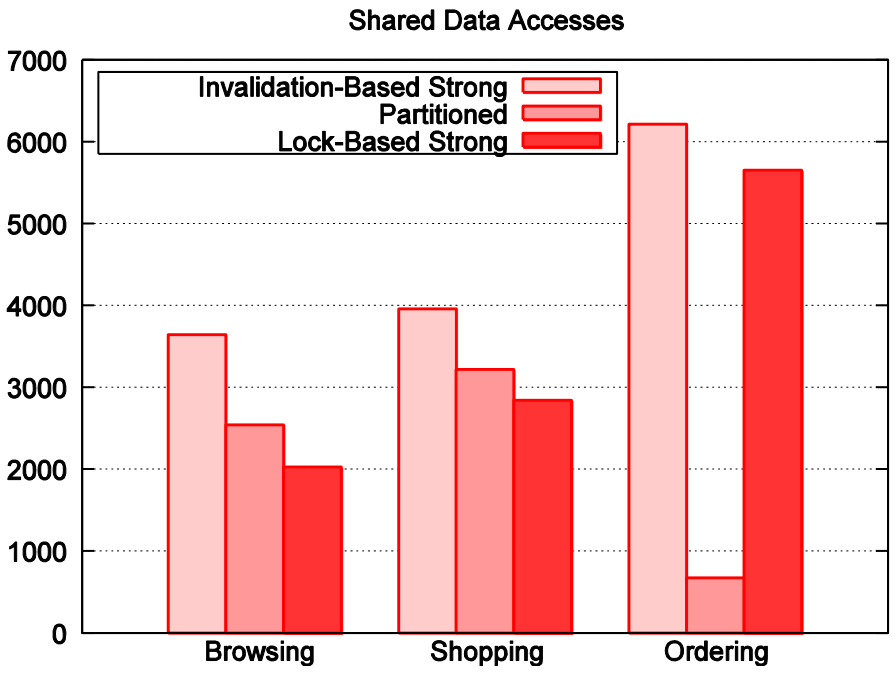
- TPC-W transactional web commerce benchmark
- Emulates an online bookstore
- 20 minutes, 144000 customers, 10000 items
- Four servers (Tomcat, MySQL)
- Three workload mixes:
  - Browsing: 95% browsing, 5% ordering
  - Shopping: 80% browsing, 20% ordering
  - Ordering: 50% browsing, 50% ordering

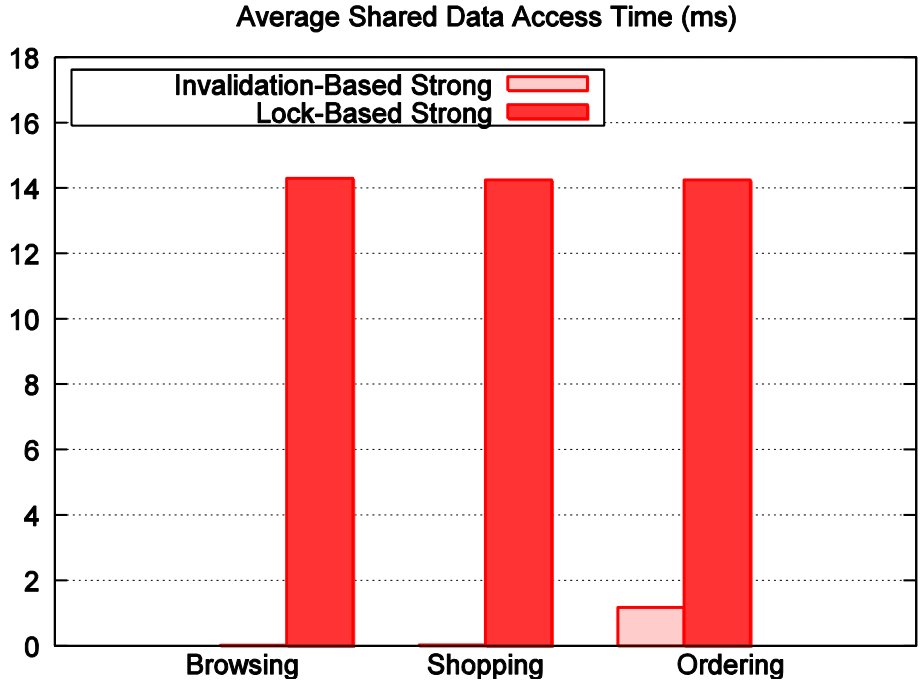
Communication Overhead



Synchronizing reads is expensive

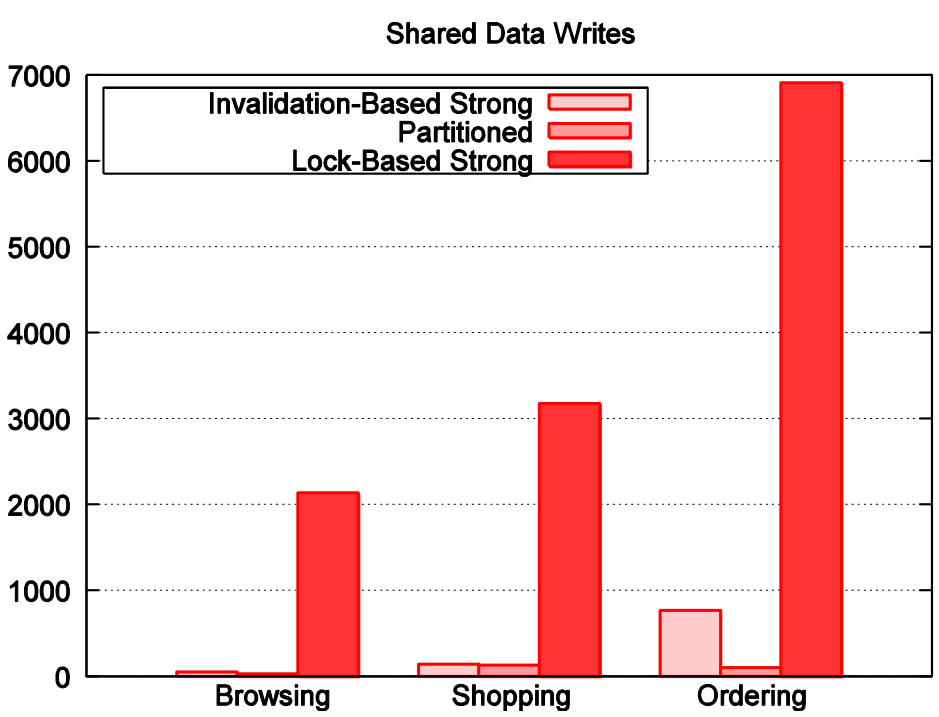
Shared Data Accesses

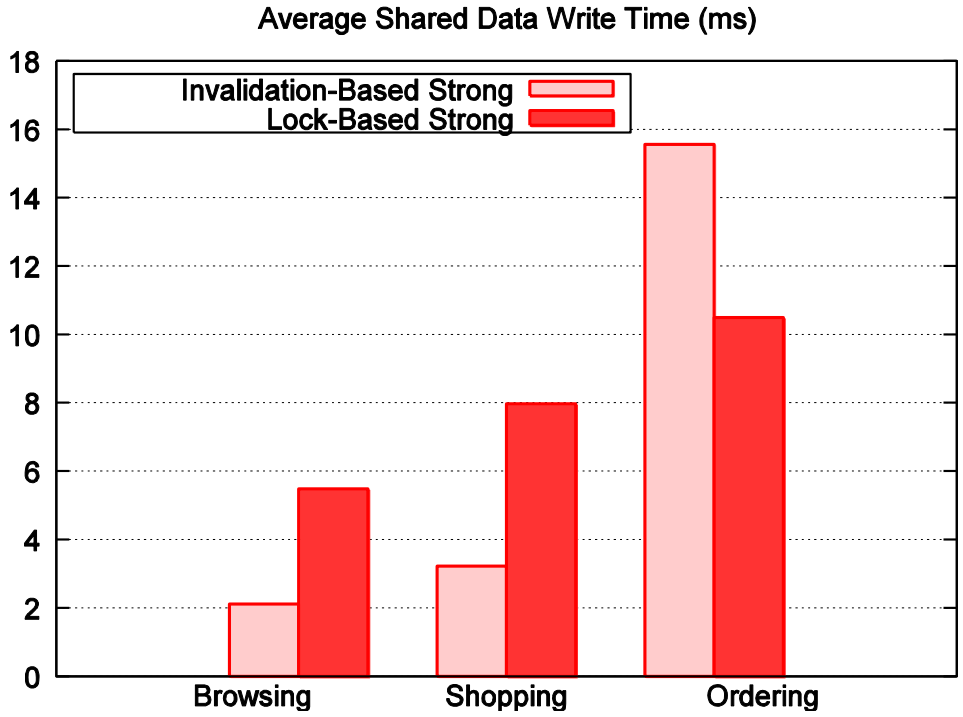




- Invalidation-based reduces access time
- Partitioning reduces shared accesses

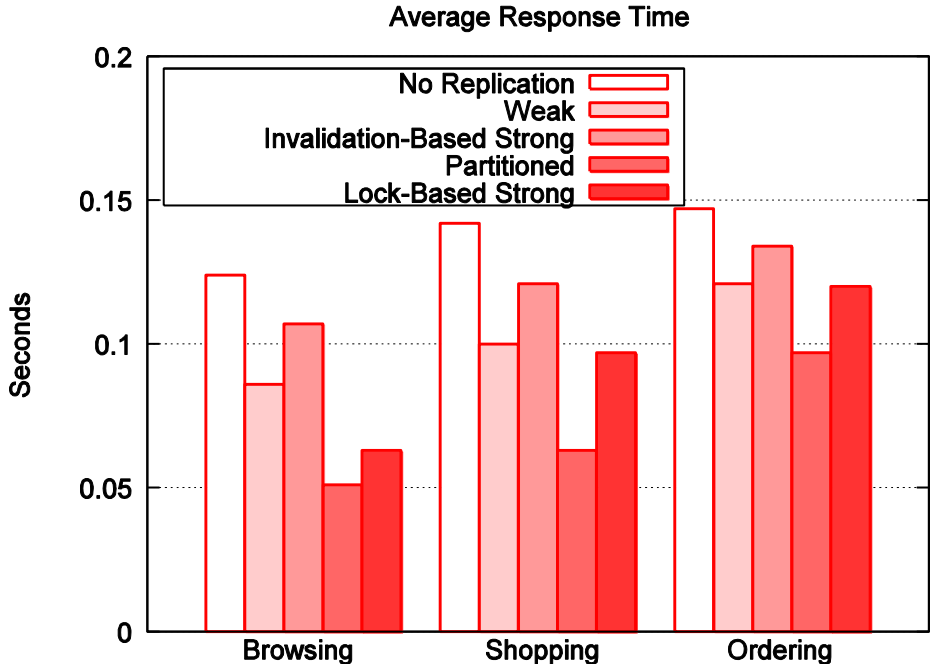
Shared Data Writes

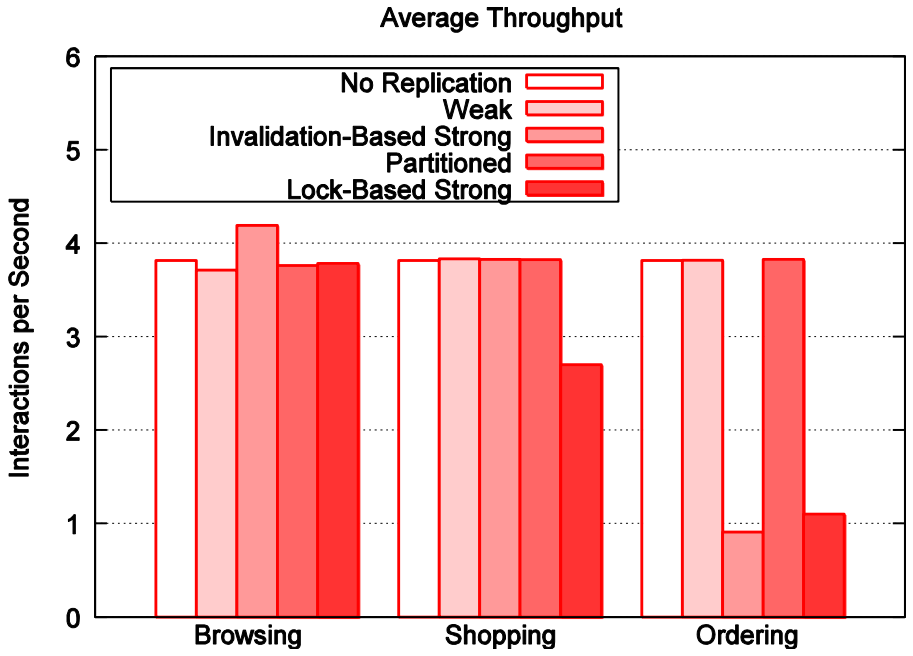




Invalidation-based shared writes are more expensive but less frequent

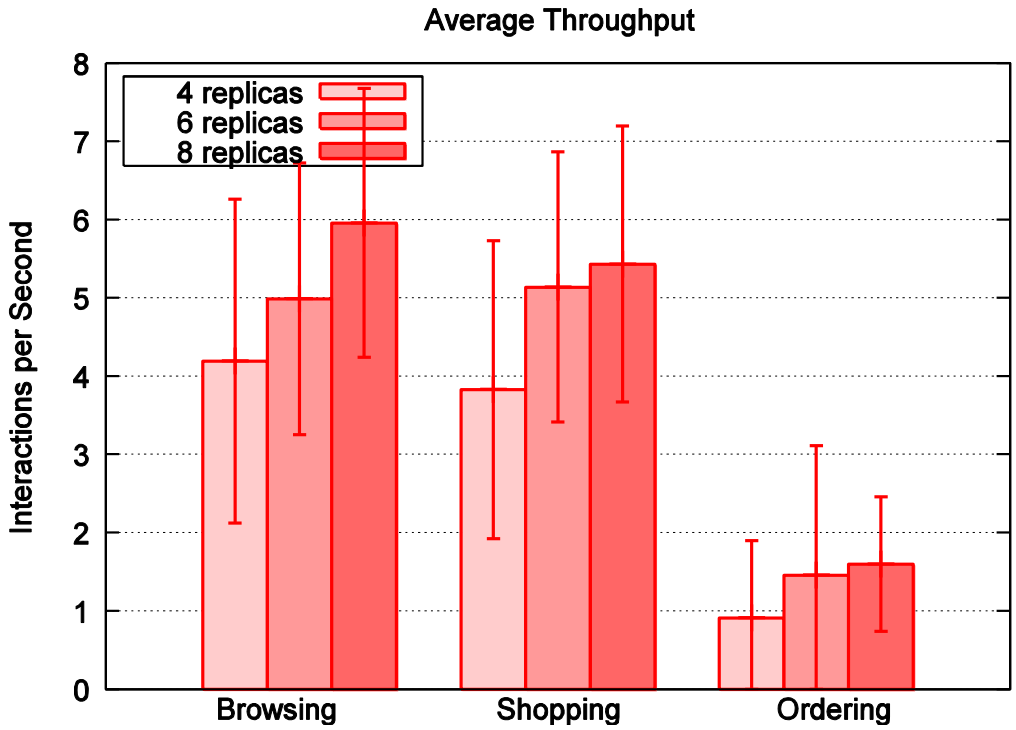
Replication Performance





- Invalidation-based not prohibitive
- Lock-based low response time but low throughput

Replication Scalability



Replication benefits read-heavy workloads more

Related Work

- Multi-Tier Architectures
  - Single data tier
  - Master-slave replication with weak consistency
  - Multi-master with group communication
  - Gossiping with weak consistency
- Distributed Databases
  - Lock-based
  - Timestamp-based
  - Optimistic with global validation or ordering
- Data Partitioning
  - Horizontal
  - Vertical

Conclusions

- Proposed an efficient, distributed, strong consistency protocol
- Implemented a replication middleware for distributed multi-tier architectures
- Studied performance of common replication approaches using the TPC-W benchmark
- Quantified the performance hit of strong consistency depending on the workload

Thank You

Thomas Repantis
Akamai Technologies
trepanti@akamai.com