Robustness in the Salus scalable block store

Yang Wang, Manos Kapritsos, Zuocheng Ren, Prince Mahajan, Jeevitha Kirubanandam, Lorenzo Alvisi, and Mike Dahlin

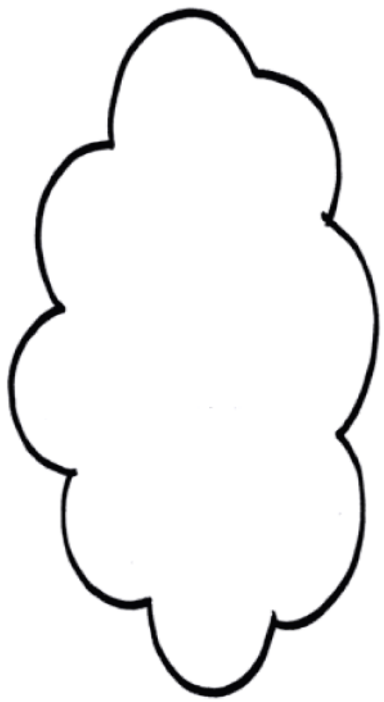
University of Texas at Austin

Storage: Do not lose data

Past

Fsck, IRON, ZFS, …

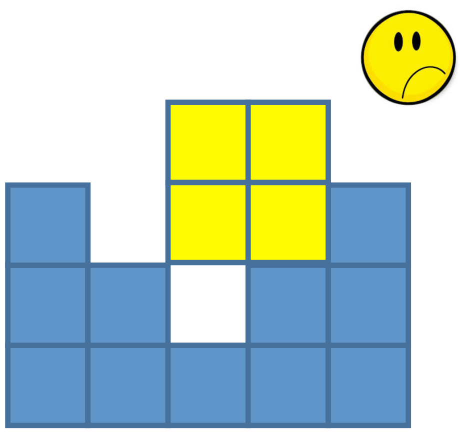
Now & Future

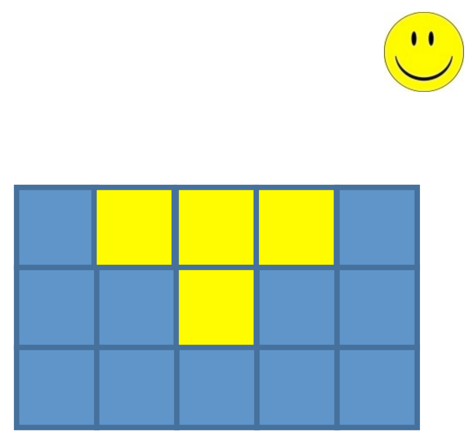


Adding robustness to scalable systems

Strong protections: Arbitrary failure
(BFT, End-to-end verification, …)

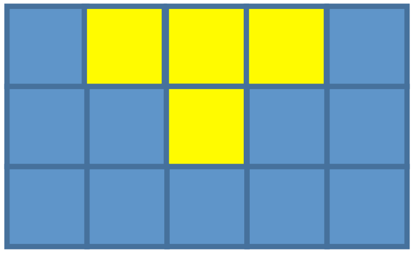
Salus





Scalable systems: Crash failure + certain disk corruption
(GFS/Bigtable, HDFS/HBase, WAS, …)

Salus: A robust and scalable block store



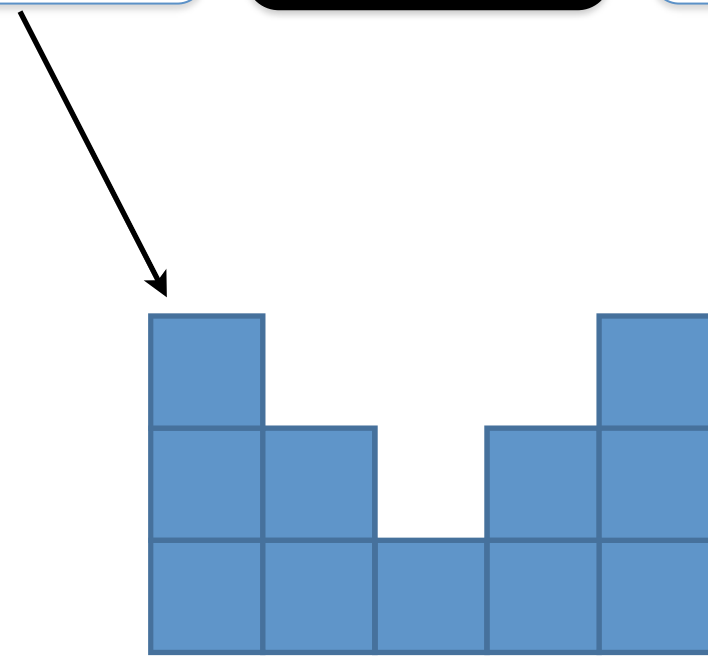
Maintain scalability

Strong robustness
- Clients never read corrupted data.
- Durable and available despite f failures.

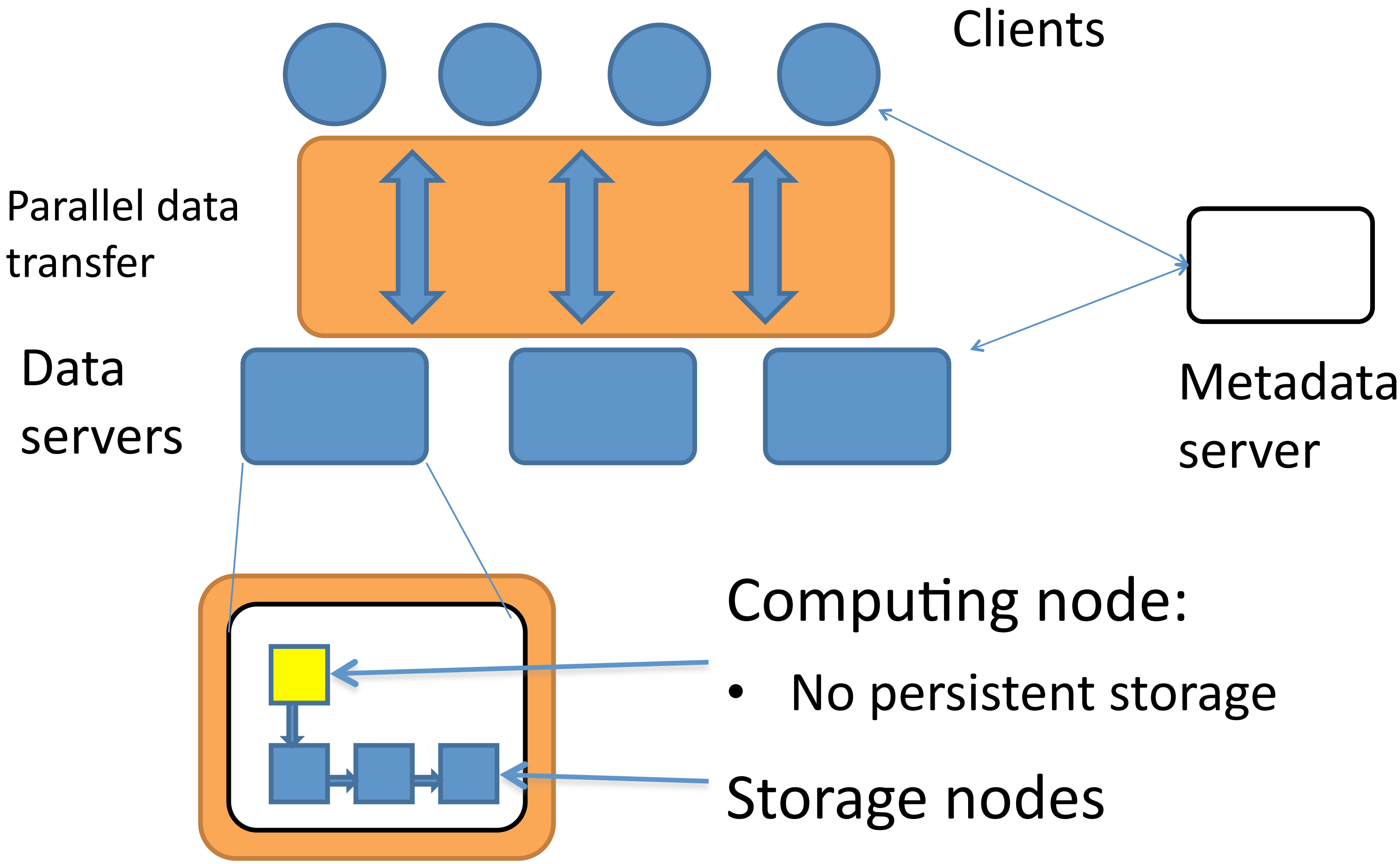
Low overhead
- Better performance in certain cases
- Comparable performance in other cases

Start from scalable key-value store

File system | Key-value | Block store | Database



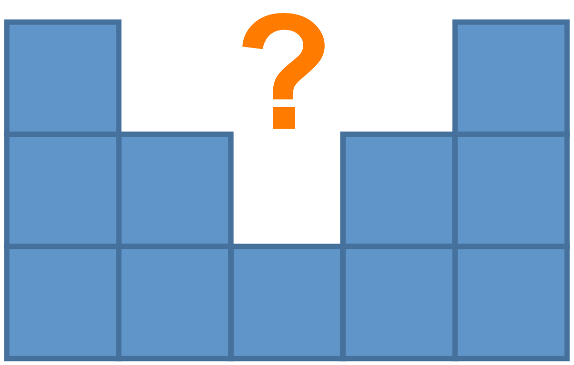
Scalable architecture



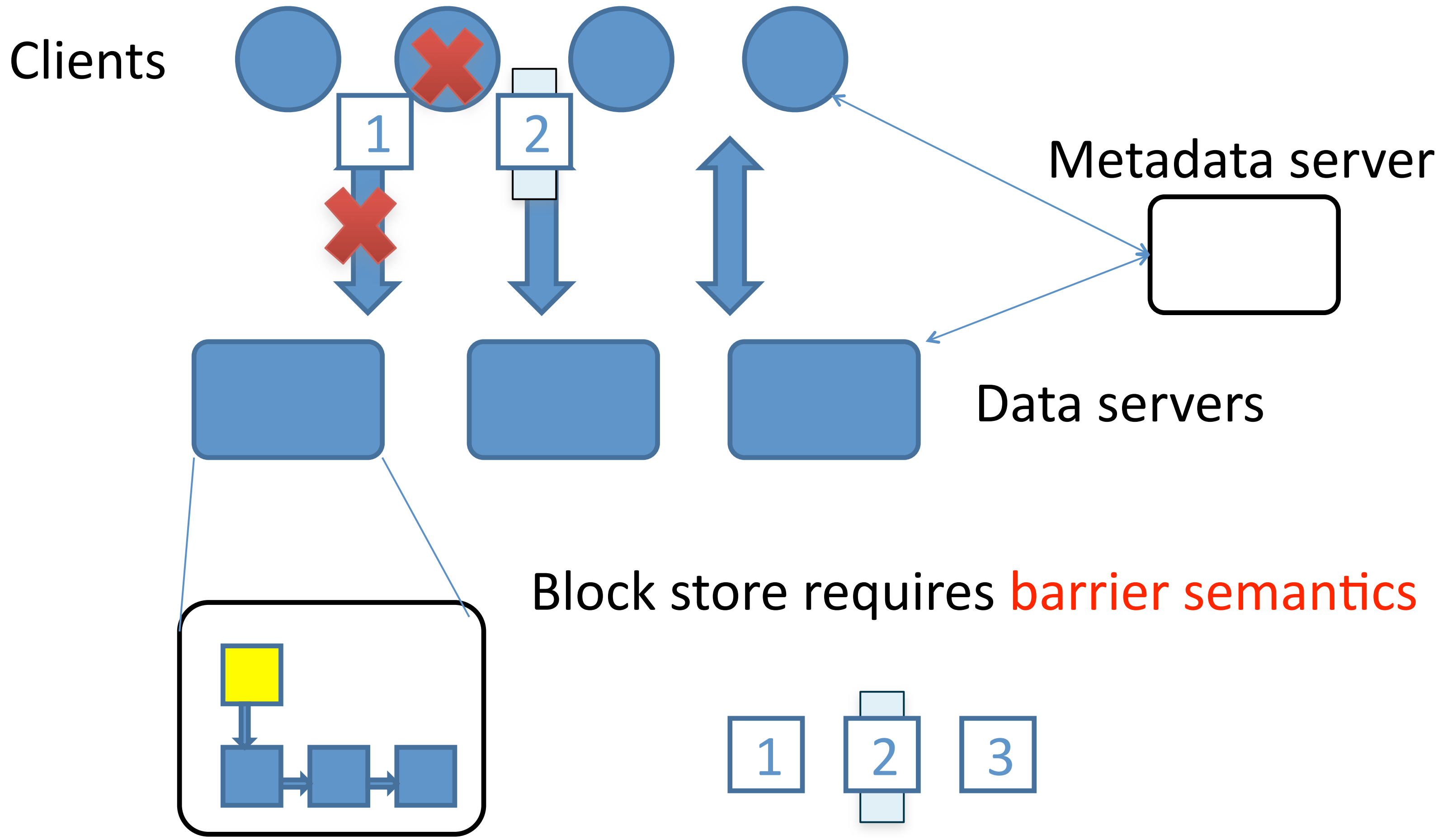
Data is replicated for durability and availability

Problems

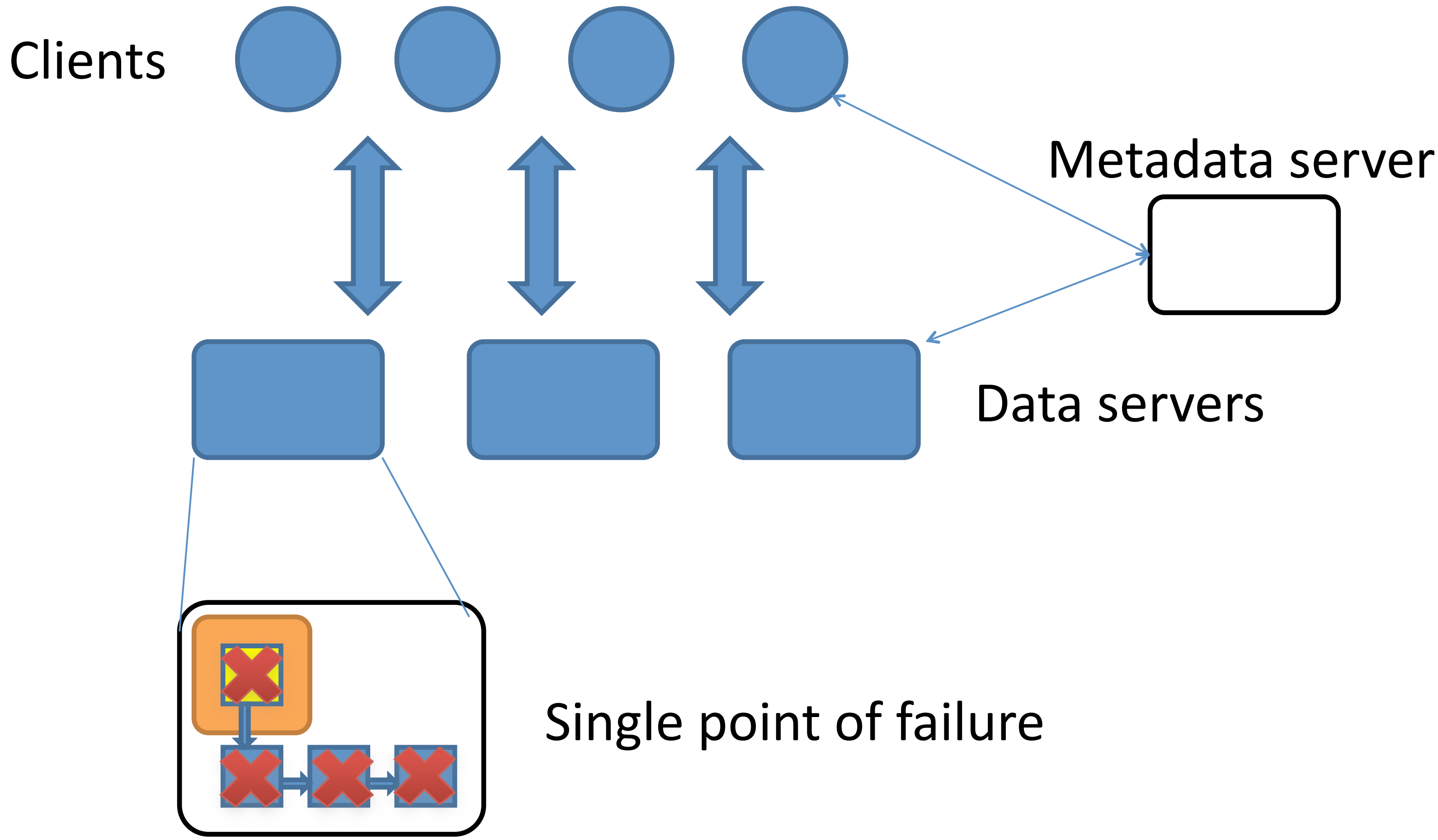
- No ordering guarantees for writes
- Single points of failures:
  - Computing nodes can corrupt data.
  - Client can not verify data integrity.



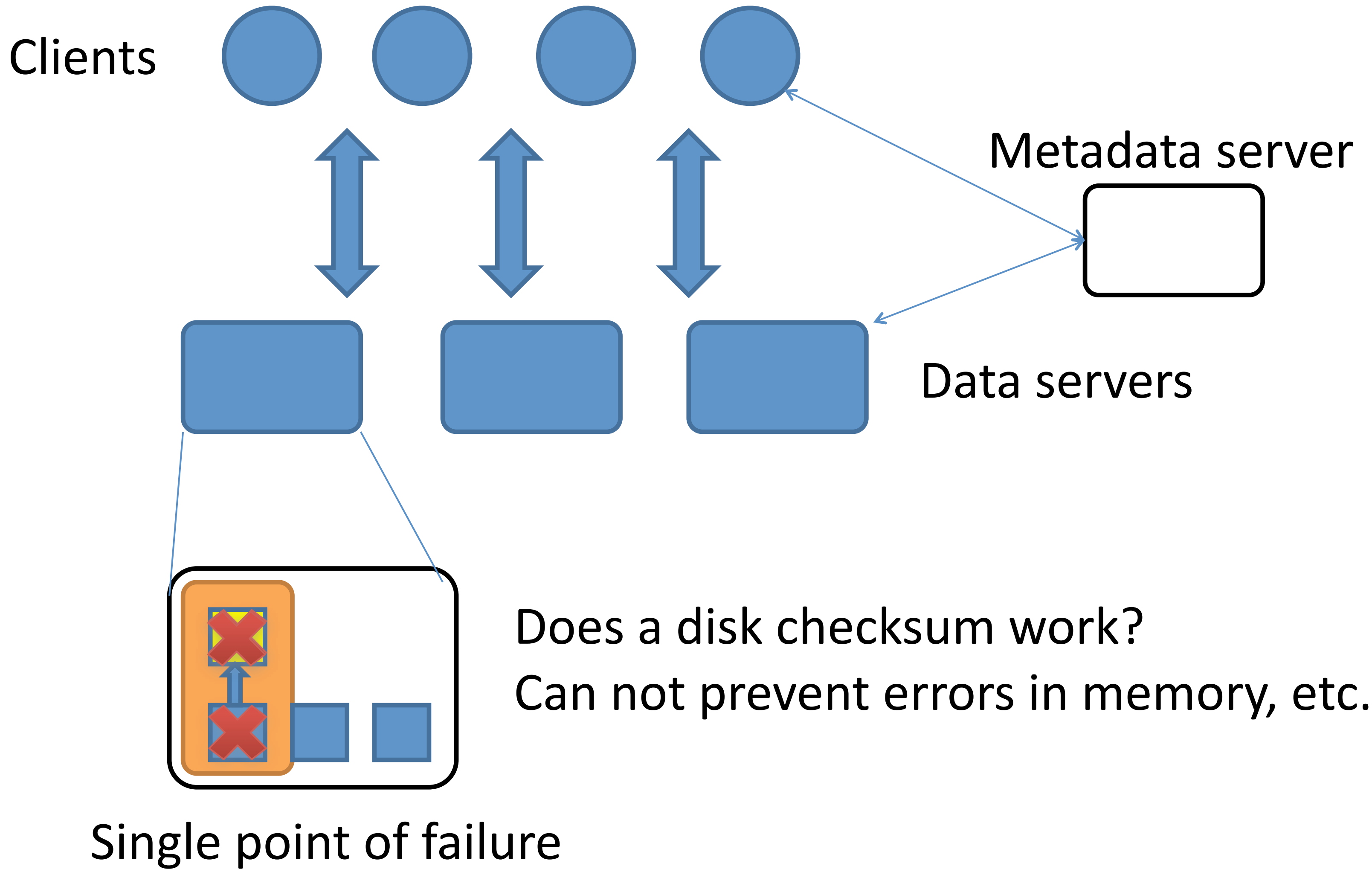
1. No ordering guarantees for writes



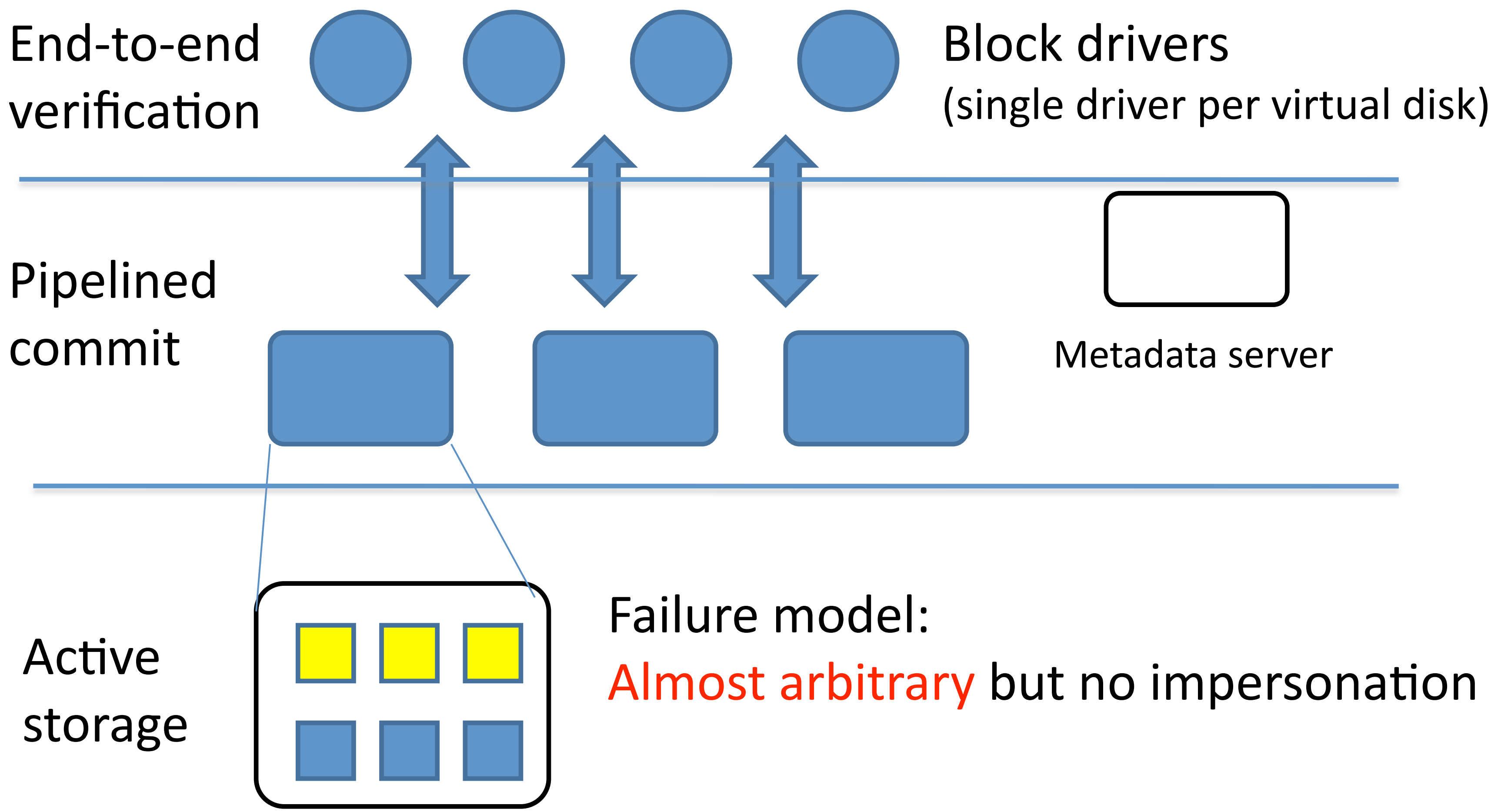
2. Computing nodes can corrupt data



3. Client can read corrupted data



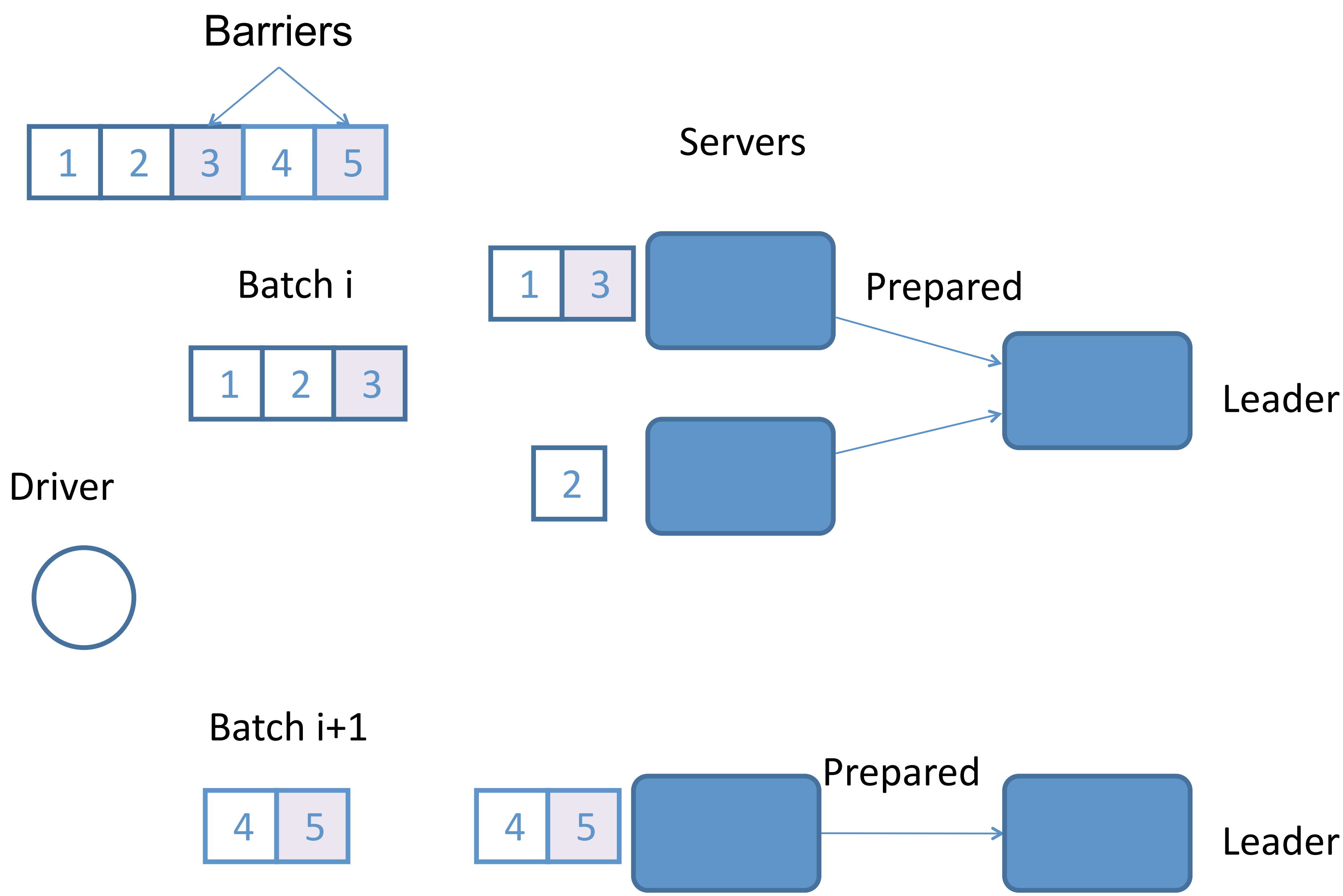
Salus: Overview



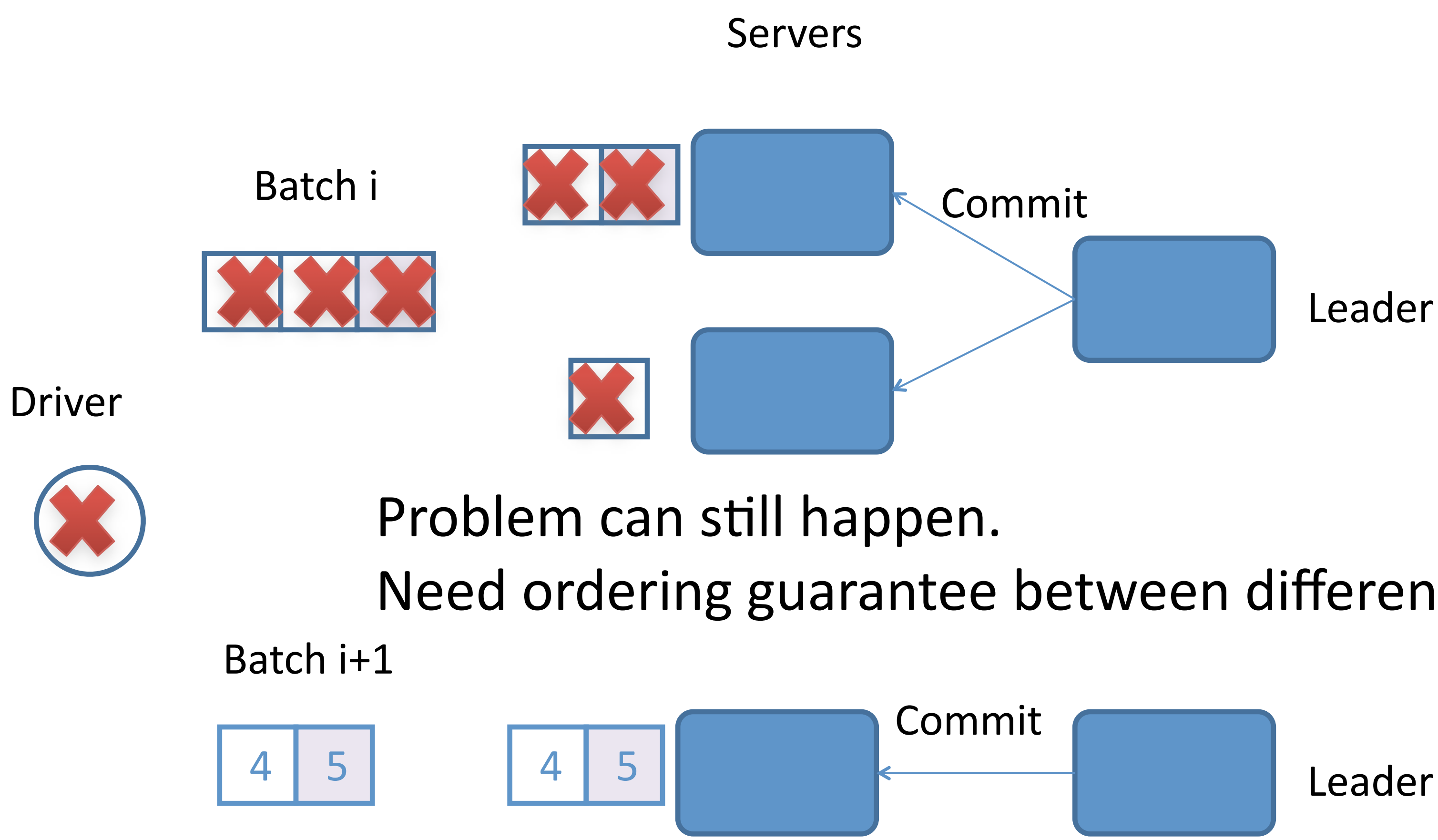
Pipelined commit

- Goal: barrier semantics
- Naïve solution:
  - Waits at a barrier: Lose parallelism
- Look similar to distributed transaction
  - Well-known solution: Two phase commit (2PC)

Problem of 2PC



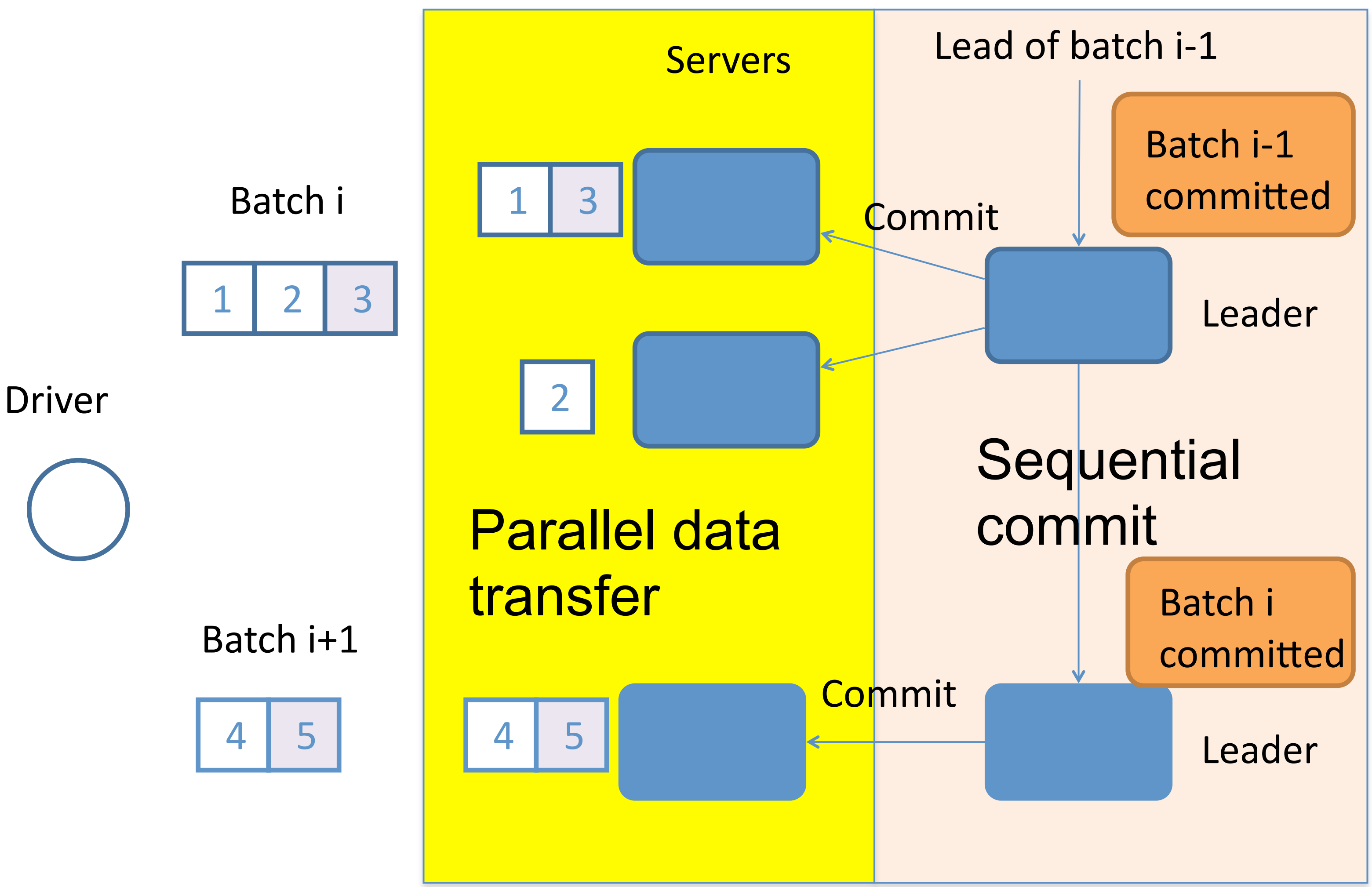
Problem of 2PC



Problem can still happen.

Need ordering guarantee between different batches.

Pipelined commit



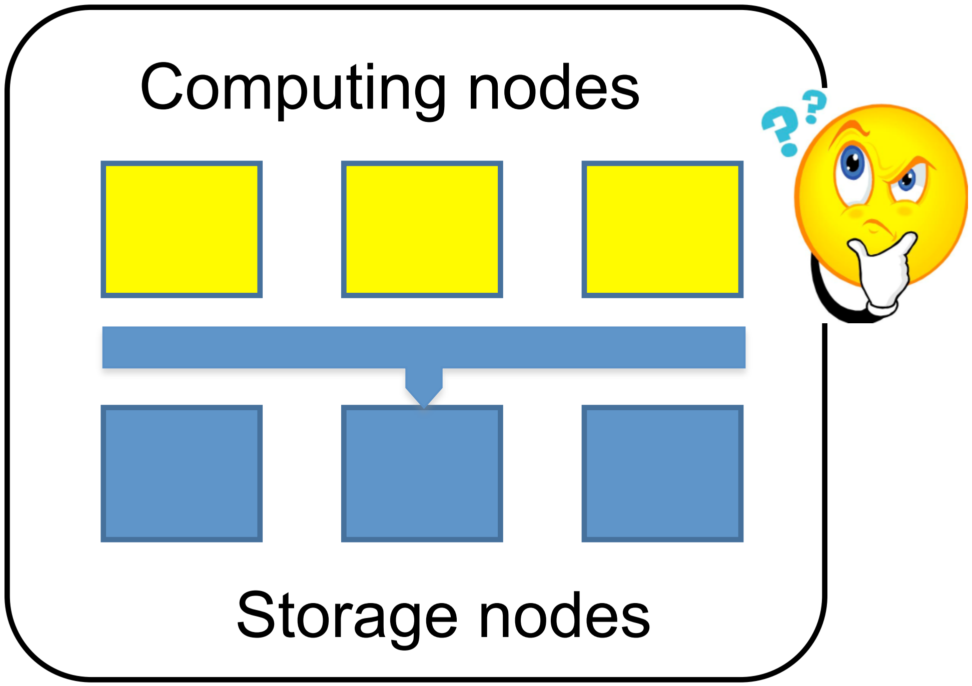
Outline

- Challenges
- Salus
  - Pipelined commit
  - Active storage
  - Scalable end-to-end checks
- Evaluation

Active storage

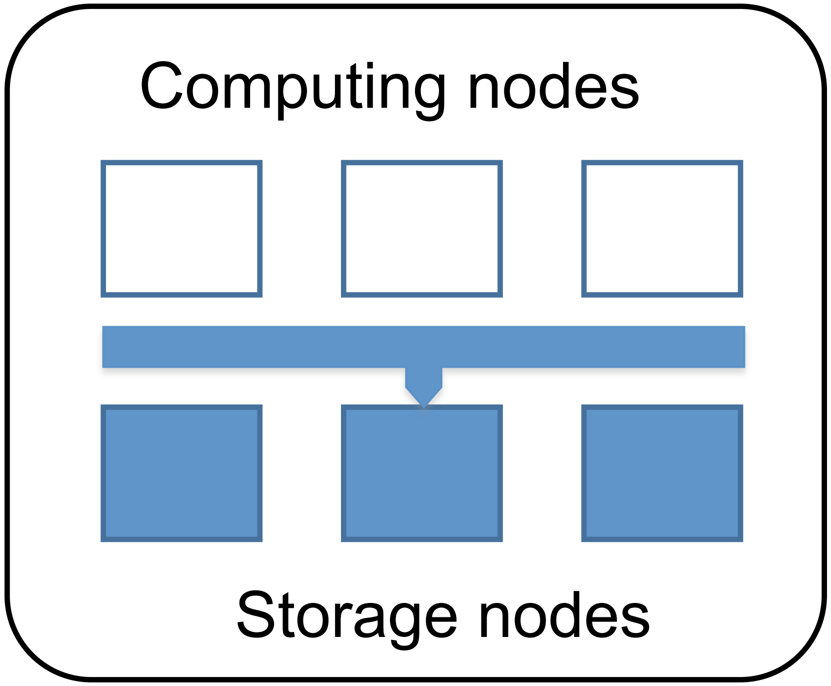
- Goal: A single node cannot corrupt data
- Well-known solution: BFT replication
  - Problem: Requires at least 2f+1 replicas
- Salus' approach: Decouple safety and liveness
  - For safety, unanimous consent of f+1 replicas
  - How about availability/liveness?

Active storage: Restore availability



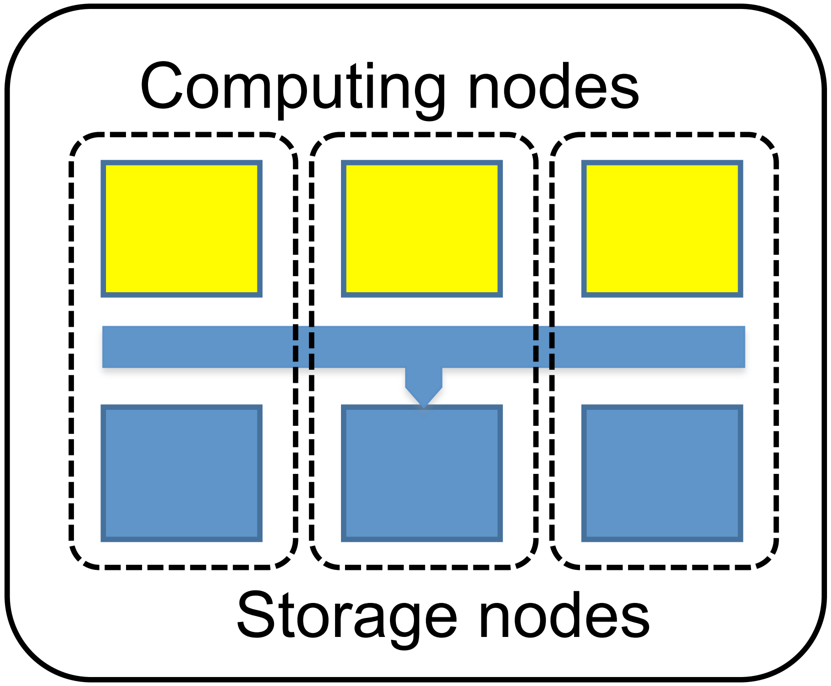
- What if something goes wrong?
  - Problem: We may not know which one is faulty.
- Replace all computing nodes
  - Computing nodes have only soft states.

Active storage: Restore availability



- What if something goes wrong?
  - Problem: We may not know which one is faulty.
- Replace all computing nodes
  - Computing nodes have only soft states.

Active storage: Better performance



- Additional benefit:
  - Colocate computing and storage: Save network bandwidth

Outline

- Challenges
- Salus
  - Pipelined commit
  - Active storage
  - Scalable end-to-end checks
- Evaluation

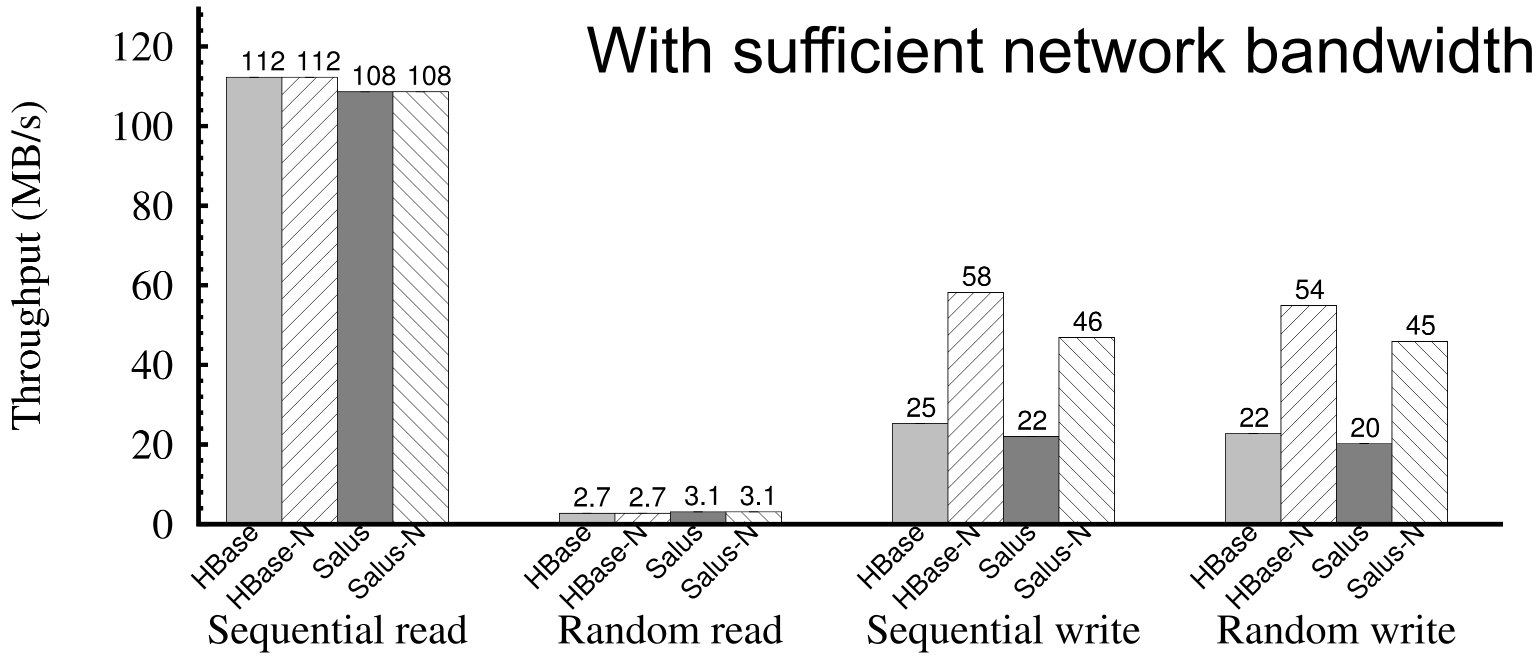
Evaluation

- Is Salus robust against failures?
- What is the overhead of Salus?
- Does the overhead grow with the scale of the system?

Is Salus robust against failures?

Failures	HBase		Salus	
	Safe	Live	Safe	Live
1-2 storage node disk failures	Yes	Yes	Yes	Yes
3 storage node disk failures	Yes	No	Yes	No
1-2 computing node memory failures	No	-	Yes	Yes
3 computing node memory failures	-	-	Yes	No

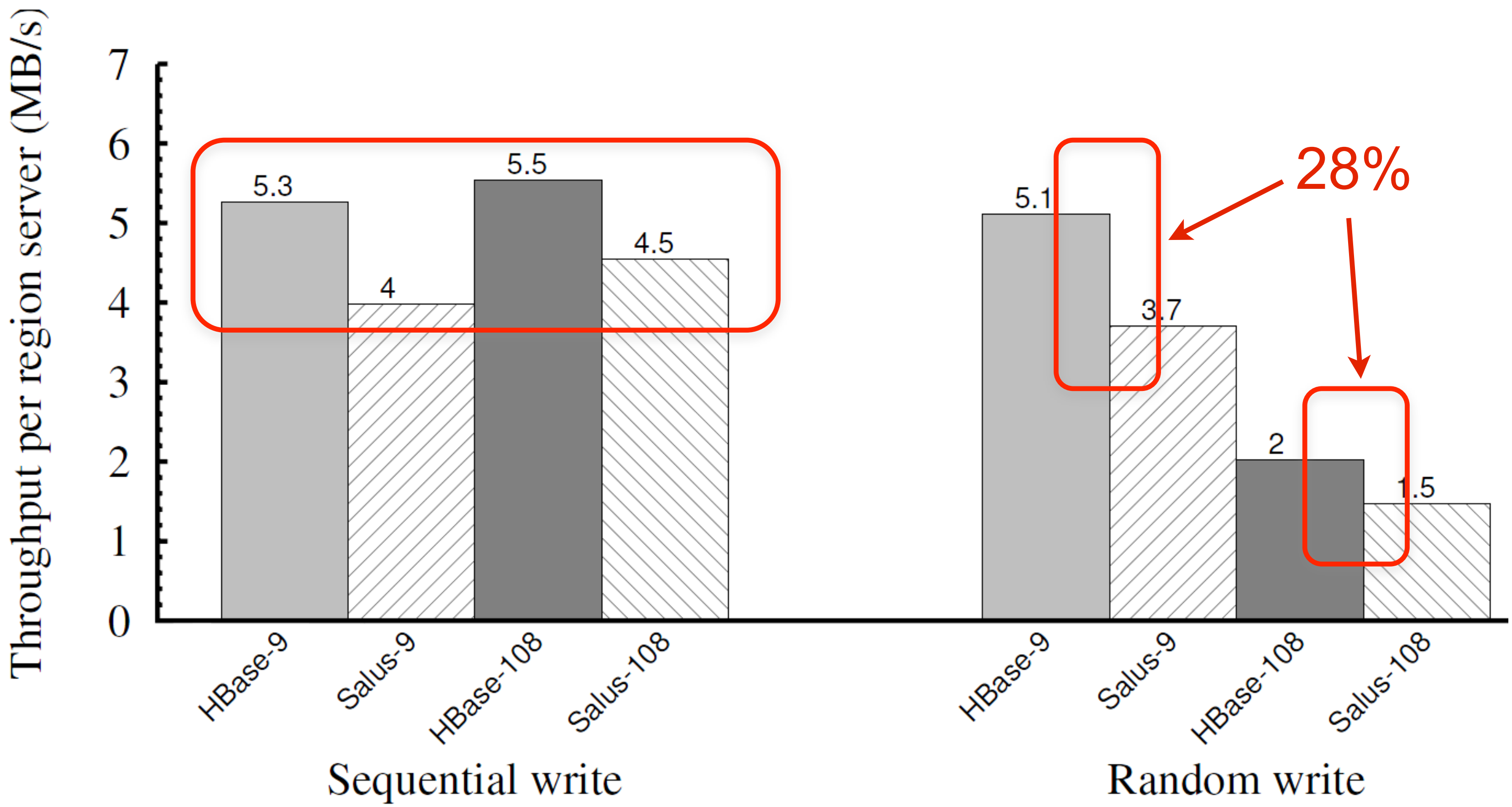
What is the overhead of Salus?



	HBase	Salus
Throughput (MB/s)	27	47
Network consumption (network bytes per byte written by the client)	5.3	2.4

With limited network bandwidth

Does the overhead grow with the scale of the system?



Conclusion

High robustness ≠ Low performance

- Pipelined commit
  - Write in parallel
  - Provide ordering guarantees
- Active Storage
  - Eliminate single point of failures
  - Consume less network bandwidth