Limplock: Understanding the Impact of Limpware on Scale-out Cloud Systems

Thanh Do*, Mingzhe Hao, Tanakorn Leesatapornwongsa, Tiratat Patana-anake, and Haryadi S. Gunawi

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY of WISCONSIN MADISON

Hardware fails

- Growing complexity of …
  - Technology scaling
  - Manufacturing
  - Design logic
  - Usage
  - Operating environment
- … makes HW fail differently
  - Complete fail-stop
  - Fail partial
  - Corruption
  - Performance degradation?

} Rich literature

The 1st anecdote

Degraded NIC! (1000000x)

"… 1Gb NIC card on a machine that suddenly starts transmitting at 1 kbps,

this slow machine caused a chain reaction upstream in such a way that the performance of entire workload for a 100 node cluster was crawling at a snail's pace, effectively making the system unavailable for all [illegible] purposes."
– Borthakur of Facebook

Cascading impact!

More stories in the paper

Limpware

- Does HW degrade? Yes
  - *Limpware: Hardware whose performance degrades significantly compared to its specification*
- Is this a destructive failure mode? Yes
  - Cascading failures, no "fail in place"
- No systematic analysis on its impact

Study Summary

- 56 experiments that benchmark 5 systems
  - Hadoop, HDFS, Zookeeper, Cassandra, HBase
  - 22 protocols
  - 8 hours under normal scenarios
  - 207 hours under limpware scenarios
- Unearth many limpware-intolerant designs

Our findings:

A single piece of limpware (e.g. NIC) causes severe impact on a whole cluster

Outline

- Introduction
- **System analysis**
- Limplock
- Limpware-Tolerant Systems
- Conclusion

Anecdotal impacts

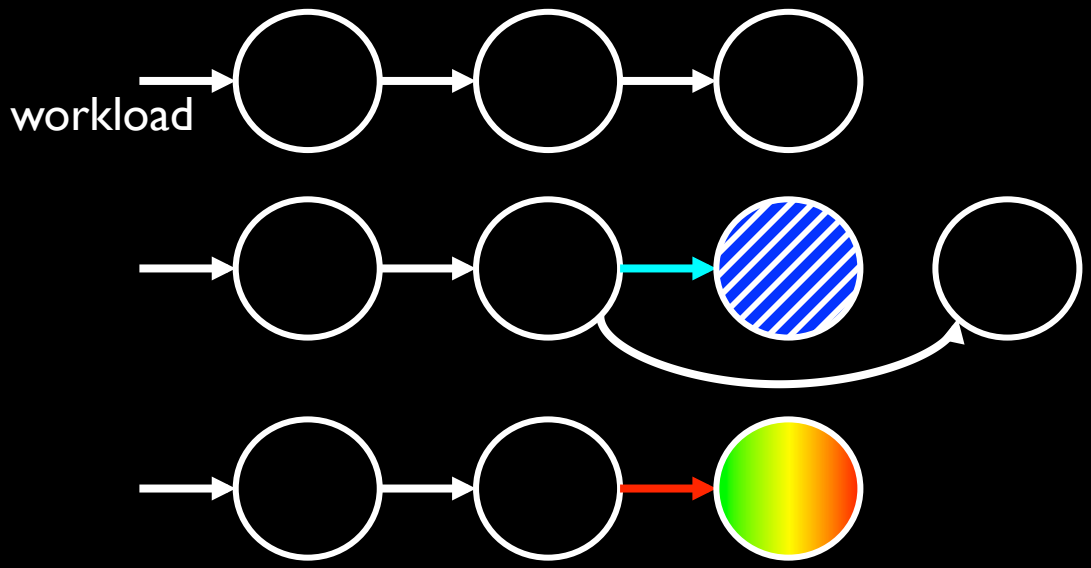
- *"The performance of a 100 node cluster was crawling at a snail's pace"* – Facebook
- But, … **why?**

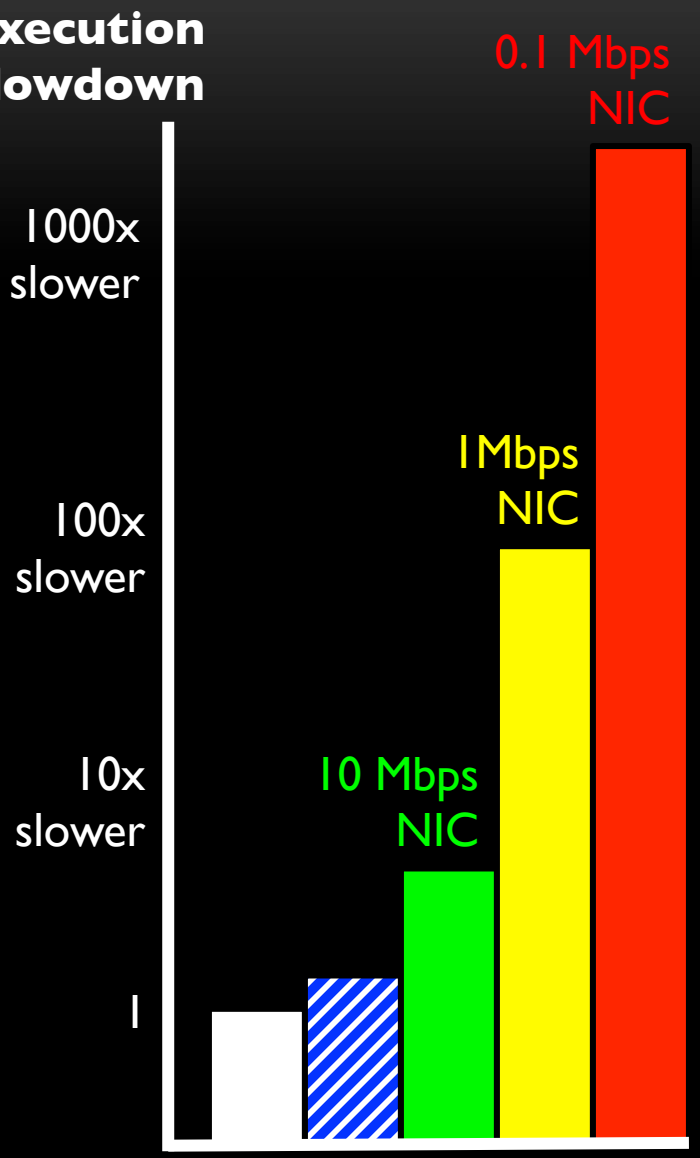
System analysis

- Goals
  - Measure system-level impacts
  - Find design flaws
- Methodology
  - Target cloud systems (e.g., HDFS, Hadoop, ZooKeeper)
  - Inject load + limpware
    - E.g. slow a NIC to 1 Mbps, 0.1 Mbps, etc.
  - White-box analysis (internal probes)
    - Find design flaws

Example

- Run a distributed protocol
  - E.g., 3-node write in HDFS
- Measure slowdowns under:
  - No failure, crash, a degraded NIC



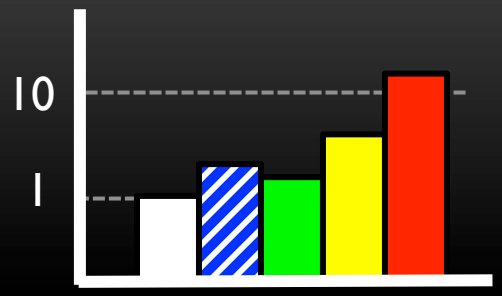


Outline

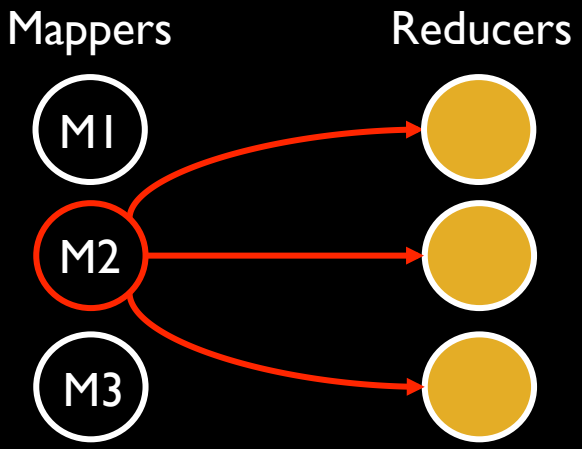
- Introduction
- **System analysis**
  - **Hadoop case study**
- Limplock
- Limpware Tolerant Cloud Systems
- Conclusion

Hadoop Spec. Exec.

- Hadoop tail-tolerant?
  - Why speculative exec is not triggered?
- Consider degraded NIC on a map node
  - Task M2's speed = M1 and M3
  - Input data is local!
- But all reducers are slow
  - Straggler: slow vs. others of same job
  - No straggler detected!
- **Flaws**
  - Task-level straggler detection
  - Single point of failure!

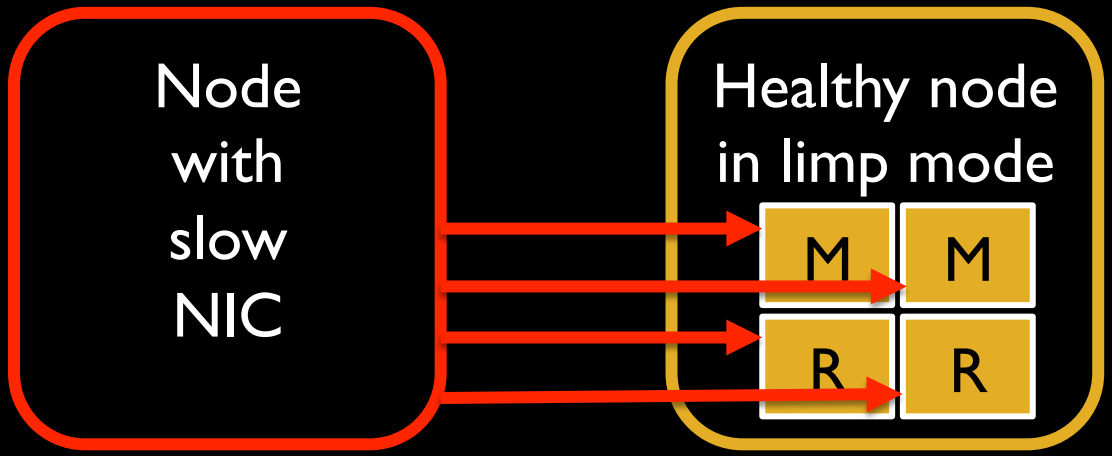


Wordcount on Hadoop



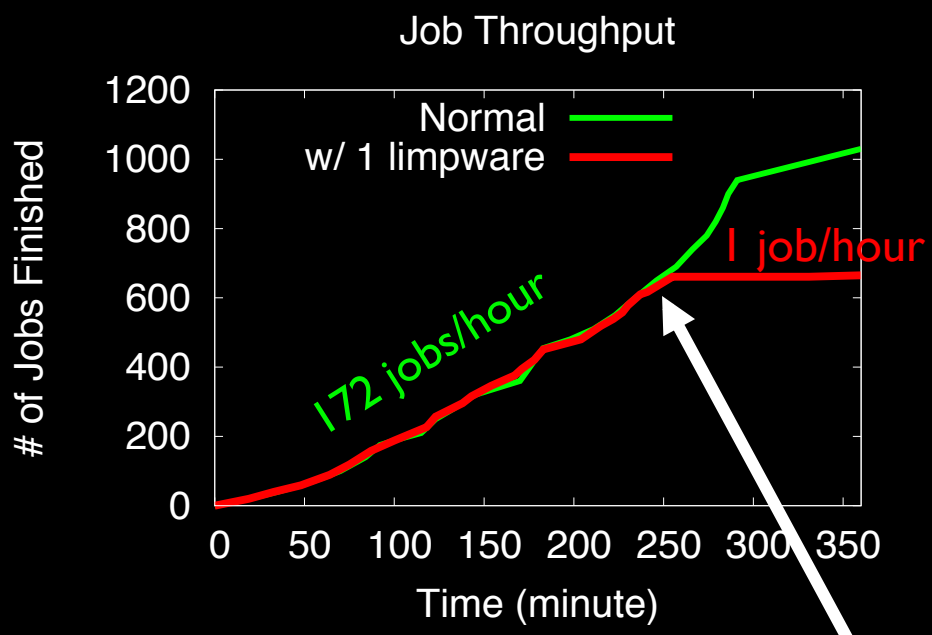
Cascading failures

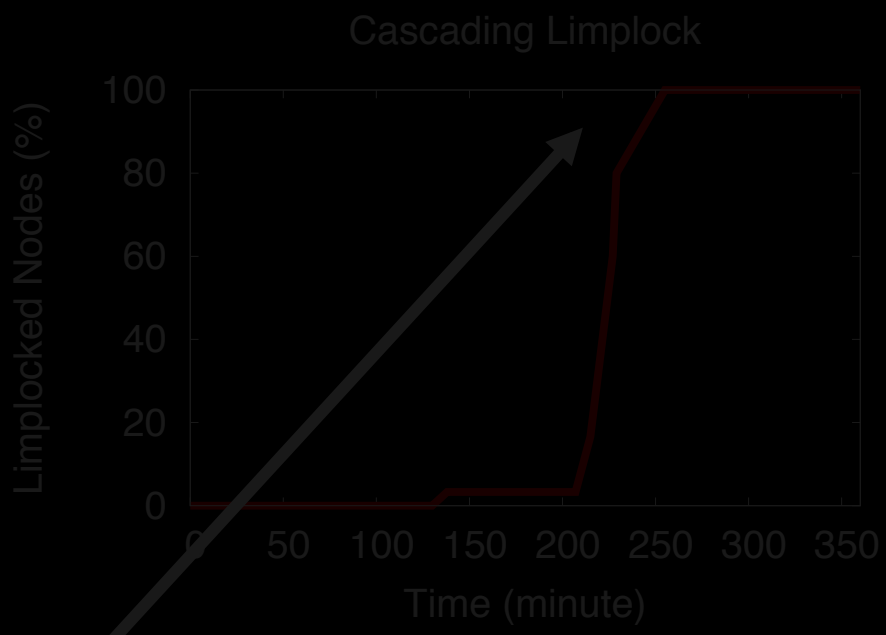
- A degraded NIC → degraded tasks
  - (Degraded tasks are slower by orders of magnitude)
- Slow tasks use up slots → degraded node
  - Default: 2 mappers and 2 reducers per node
  - If all slots are used → node is "unavailable"
- All nodes in limp mode → degraded cluster



Cluster collapse

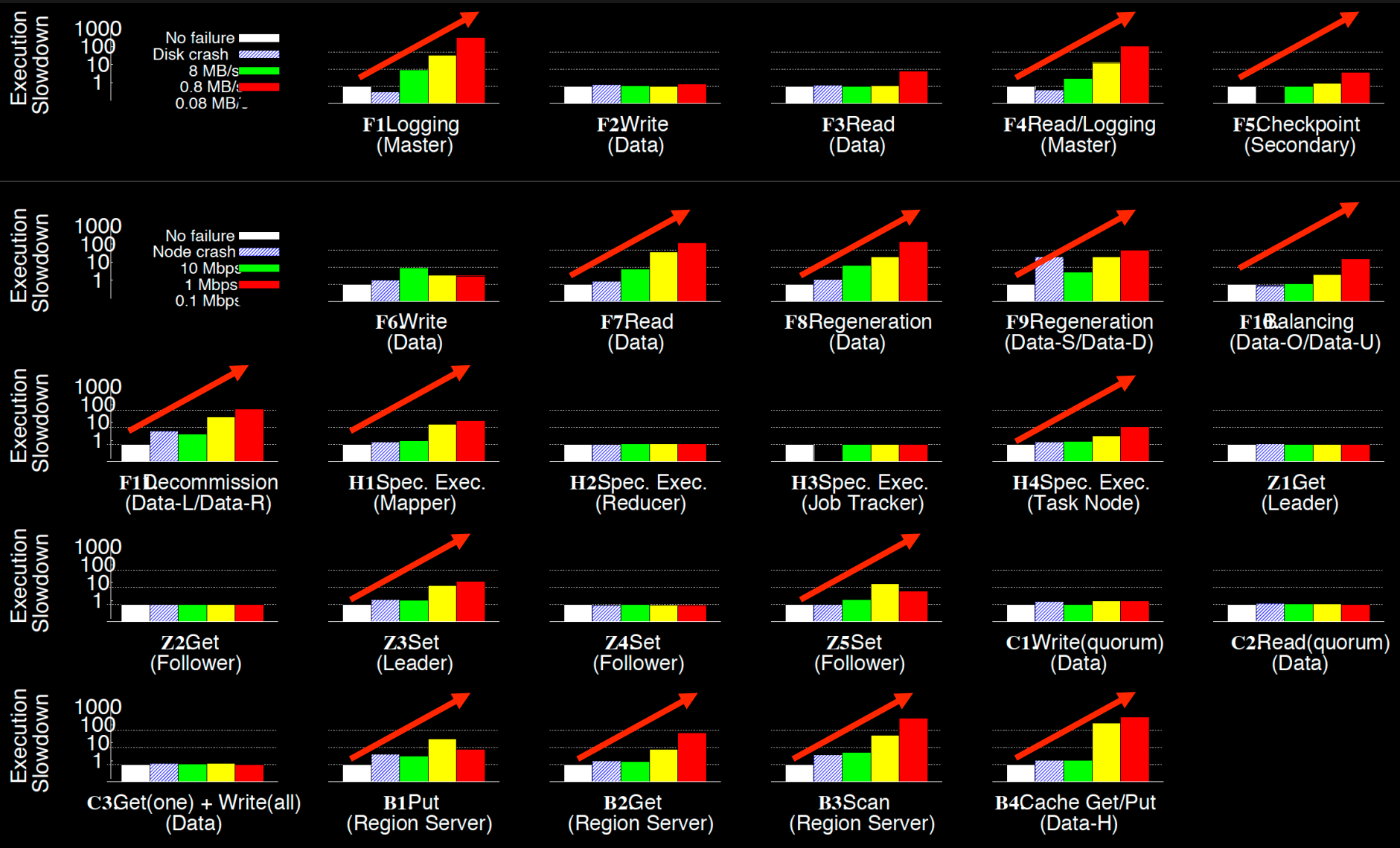
- Macrobenchmark: Facebook workload
  - **30-node cluster**
  - One node w/ degraded NIC (0.1 Mbps)





Cluster collapse!
Why?

Fail-stop tolerant, but not limpware tolerant (no failover recovery)



Outline

- Introduction
- System analysis
- Formalizing the problem: **Limplock**
  - **Definitions and causes**
- Limpware-Tolerant Systems
- Conclusion

Limplock

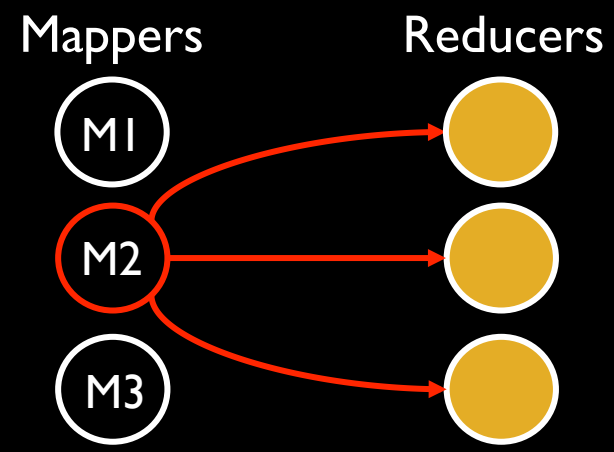
- Definition
  - The system progresses slowly due to limpware and is not capable of failing over to healthy components
  - (*i.e.*, the system is "locked" in limping mode)
- **3 levels** of limplock
  - Operation
  - Node
  - Cluster

Limplock Levels

- Operation Limplock
  - Operation involving limpware is "locked" in limping mode; no failover
- Node Limplock
  - A situation where operations that must be served by this node experience limplock, *although* the operations do *not* involve limpware
- Cluster Limplock
  - The **whole cluster** is in limplock due to limpware

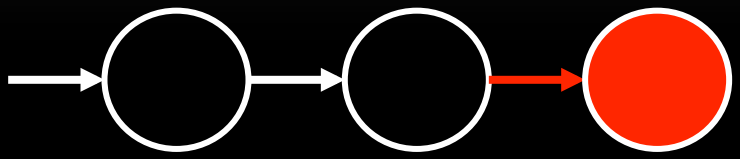
Causes of Limplock

- Operation Limplock
  - Single point of failure
    - Hadoop slow map task
    - HBase "Gateway"

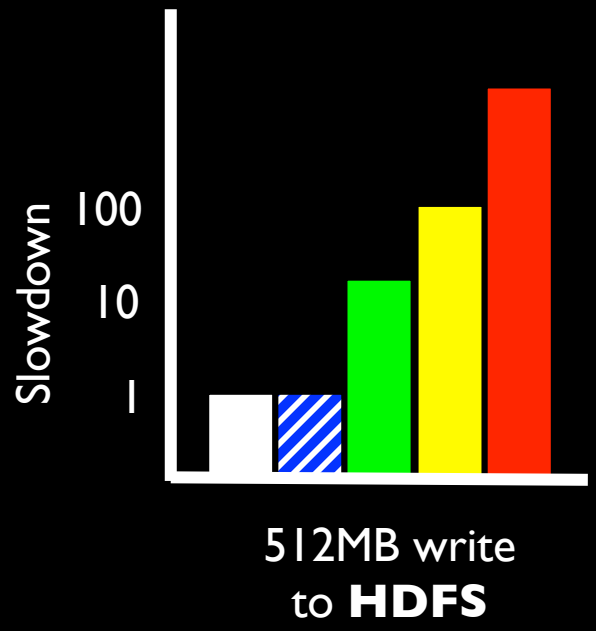


Causes of Limplock

- Operation Limplock
  - Single point of failure
  - Coarse-grained timeout
  - (more in the paper)

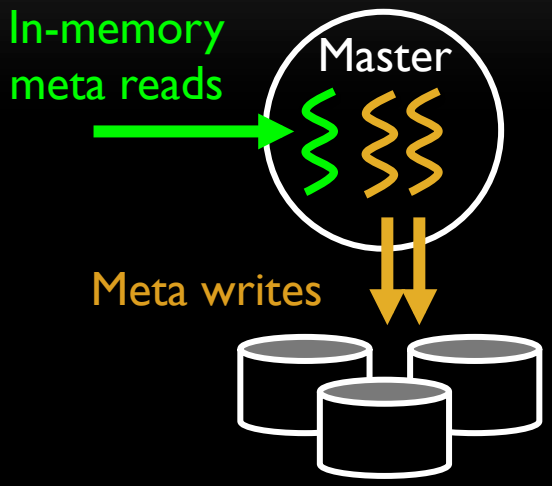


Reason: No timeout is triggered
Coarse-grained timeout in HDFS
60 second timeout on every 64 KB
Could limp almost to 1 KB/s

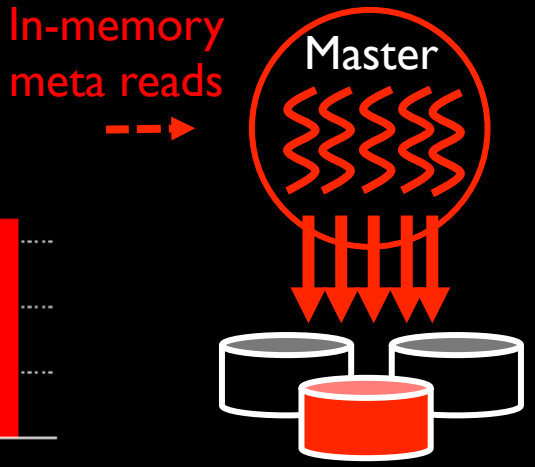


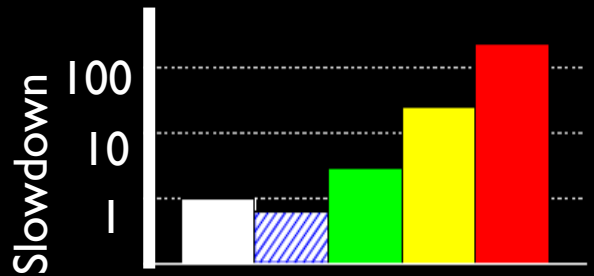
Causes of Limplock

- Operation Limplock
  - Single point of failure
  - Coarse-grained timeout
  - …
- Node Limplock
  - Bounded multi-purpose thread pool



Resource exhaustion by limplocked operation
In-memory metadata reads are blocked

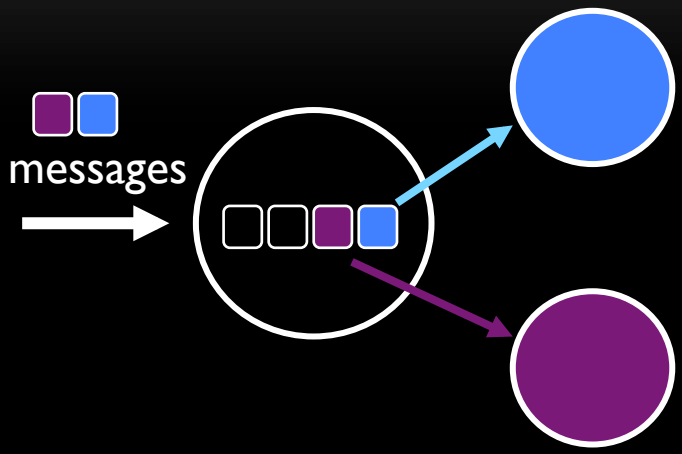


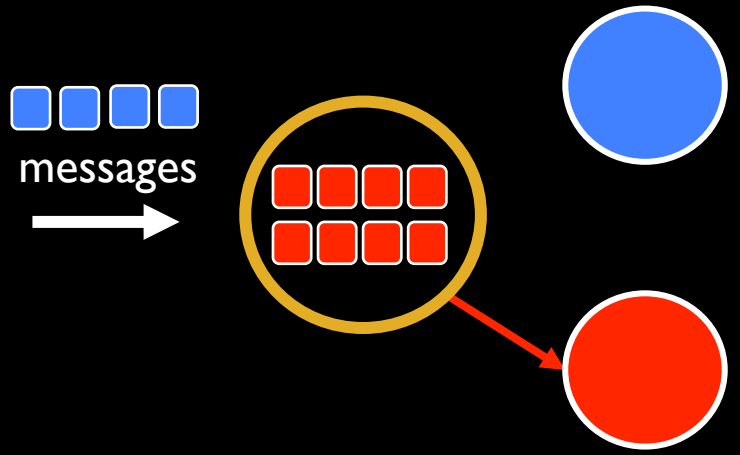


In-memory reads > 100x slower than normal

Causes of Limplock

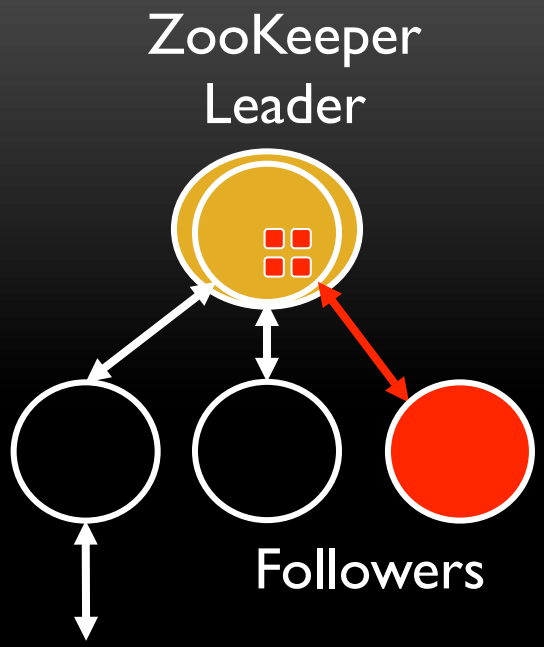
- Operation Limplock
  - Single point of failure
  - Coarse-grained timeout
  - …
- Node Limplock
  - Bounded multi-purpose thread pool
  - Bounded multi-purpose queue

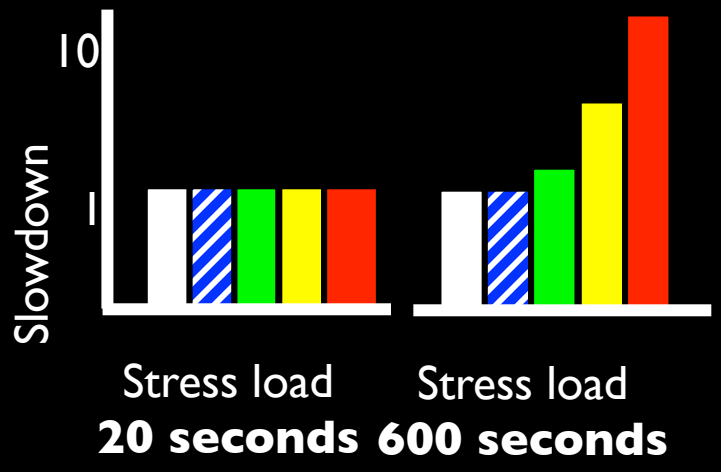




Causes of Limplock

- Operation Limplock
  - Single point of failure
  - Coarse-grained timeout
  - …
- Node Limplock
  - Bounded multi-purpose thread pool
  - Bounded multi-purpose queue
  - Unbounded thread pool/queue
    - Ex: Backlogged queue at leader
    - Node limplock at leader because garbage collection works hard
    - Quorum write: 10x slowdown





Causes of Limplock

- Operation Limplock
  - Single point of failure
  - Coarse-grained timeout
  - …
- Node Limplock
  - Bounded multi-purpose thread pool
  - Bounded multi-purpose queue
  - Unbounded thread pool/queue
- Cluster Limplock
  - All nodes in limplock
    - Ex: resource exhaustion in Hadoop, HDFS Regeneration
  - Master limplock in master-slave architecture
    - Ex: cases in ZooKeeper, HDFS

Analysis Results

Limplock happens in almost all systems we have analyzed

- Found 15 protocols that exhibit limplock
  - 8 in HDFS
  - 1 in Hadoop
  - 2 in ZooKeeper
  - 4 in HBase

Outline

- Introduction
- System analysis
- Limplock
- **Limpware-Tolerant Cloud Systems**
- Conclusion

Principles of limpware …

- Anticipation
  - Limpware-tolerant design patterns
  - Limpware static analysis
  - Limpware statistics
    - Existing work: memory failure, disk failure, etc.
- Detection
  - Performance degradation → implicit (no hard errors)
  - Study explicit causes (e.g. block remapping, error correcting)
- Recovery
  - How to "fail in place"?
  - Better to fail-stop than fail-slow?
  - Quarantine?
- Utilization
  - Fail-stop: fail or working
  - Limpware: degrade 1-100%

Conclusion

- New failure modes → transform systems
- Limpware is a "new", destructive failure mode
  - Orders of magnitude slowdown
  - Cascading failures
  - No "fail in place" in current systems

A need for Limpware-Tolerant Systems

Thank you!
Questions?