Analysis of HDFS Under HBase

A Facebook Messages Case Study

Tyler Harter, Dhruba Borthakur*, Siying Dong*, Amitanand Aiyer*,
Liyin Tang*, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau

University of Wisconsin-Madison

*Facebook Inc.

Intro: Why Messages?

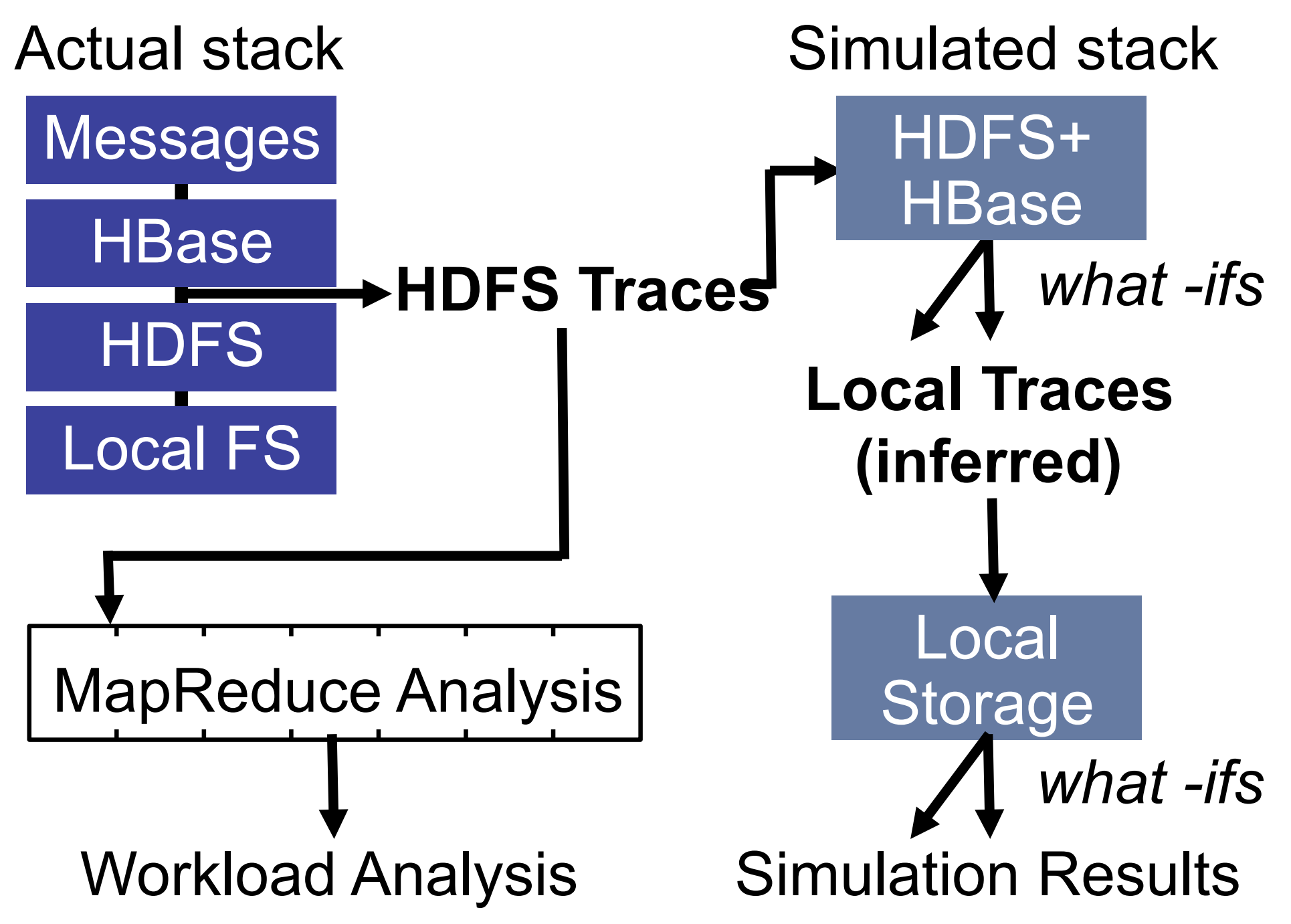
Representative application
- Backend for texts, chats, and emails

Representative of HBase over HDFS
- Used by Facebook and other companies
- Also like BigTable over GFS (Google)

Representative of layered storage
- Storage rarely "built from scratch"

Research questions
- Are Messages and MapReduce similar HDFS workloads? Is HDFS a suitable backend?
- How should Messages use flash (if at all)?
- What are the costs of layering (if any)?

Methodology
- New HDFS trace framework (open source)
- Collect traces in shadow cluster
- Analyze traces and simulate changes


Actual stack

Messages

HBase

HDFS

Local FS

HDFS Traces

MapReduce Analysis

Workload Analysis

Simulated stack

HDFS+ HBase

what -ifs

Local Traces (inferred)

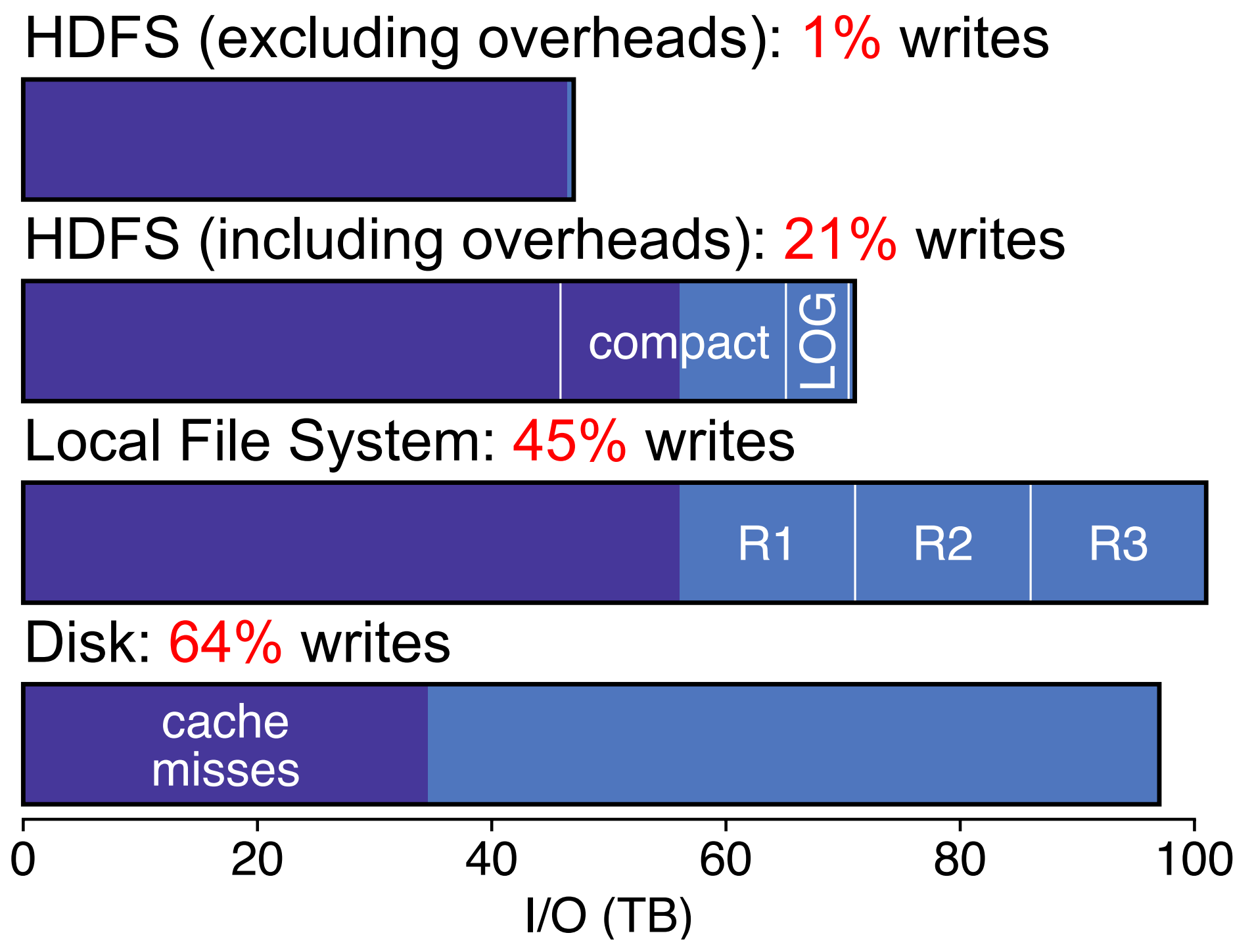
Local Storage

what -ifs

Simulation Results

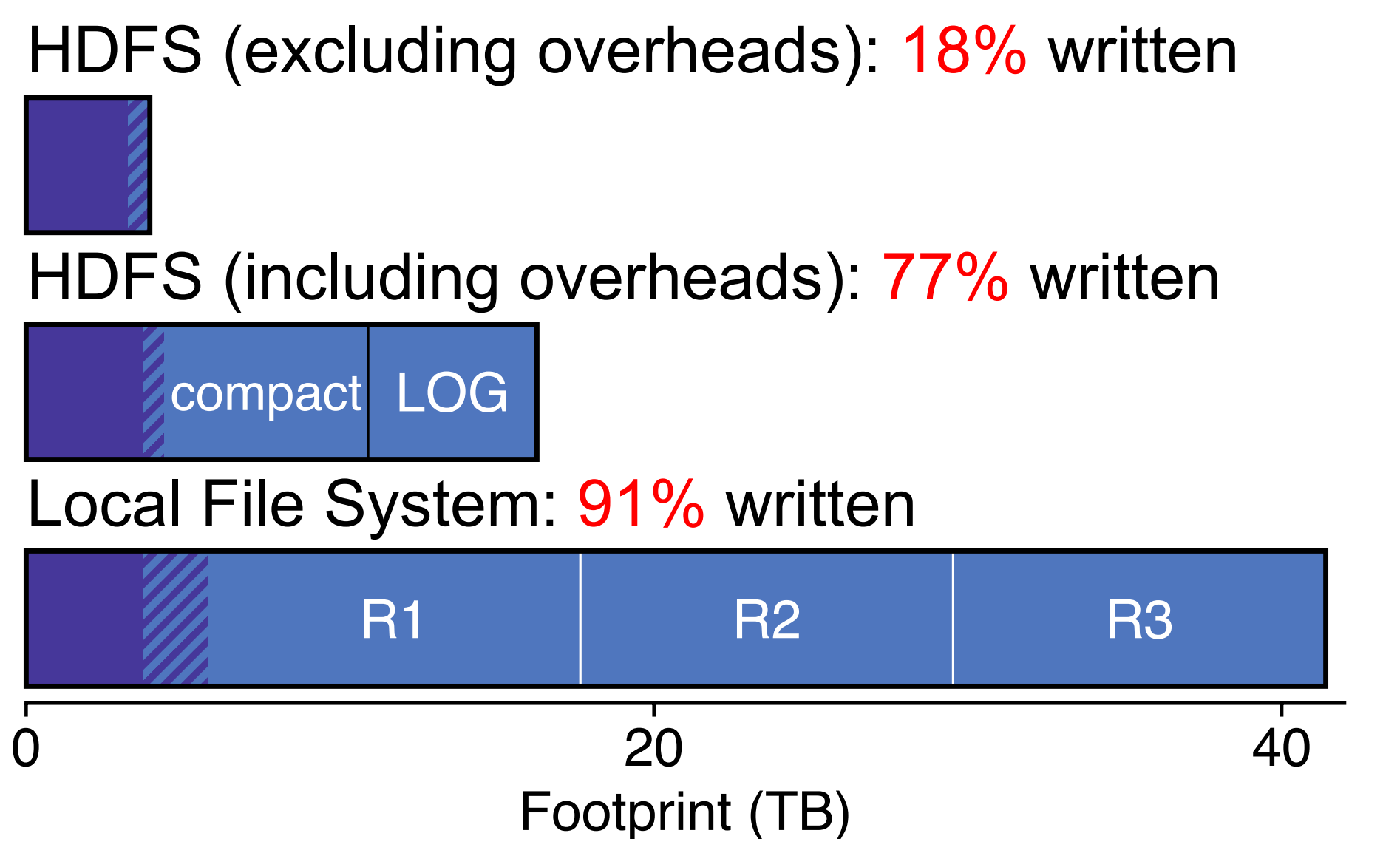
Workload Analysis

Q: What is the read/write ratio across layers?



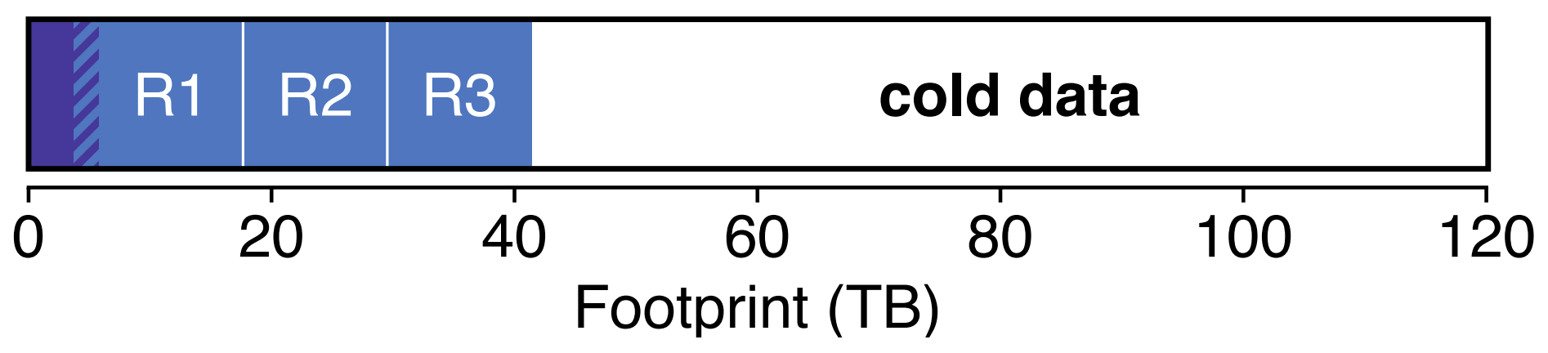
A: Layers amplify write percent from 1% to 64%

Q: How much data is touched across layers?



A: 41TB; most data is written or read (not both)

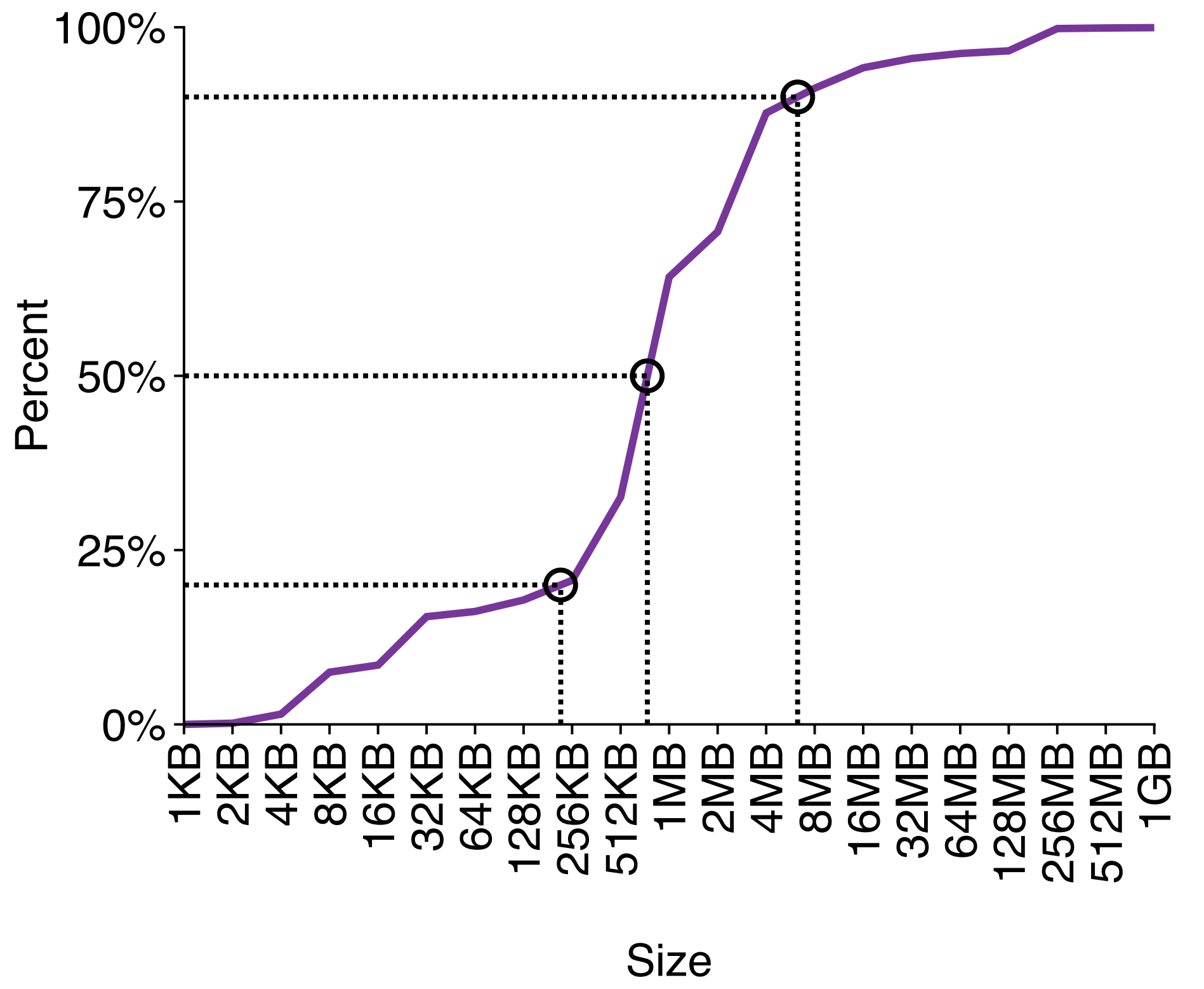
Q: How much data is cold?



A: 2/3 of the 120TB data is cold

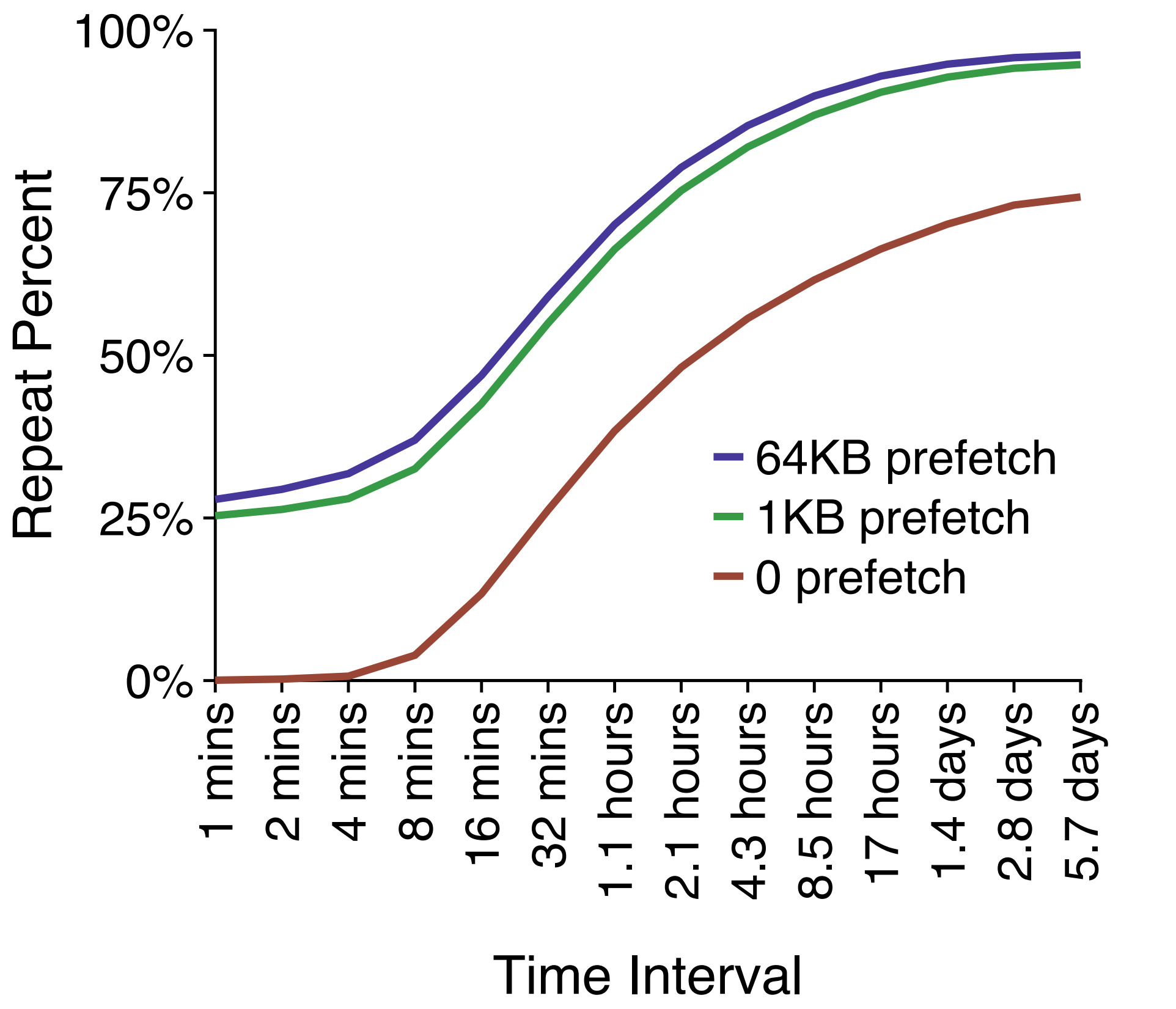
Aside: 120TB split over 9 machines is 13.3TB per machine. Storing all this in flash would be very expensive. At $0.80/GB, storing everything in flash would cost $10,895/machine.

Q: How large are created files?



A: 20% are <218KB, 50% <750KB, 90% <6.3MB

Q: What patterns are there between reads?
- *Temporal locality?*
- *Spatial locality?*
- *Sequentiality?*



A: There is significant temporal locality, suggesting additional caching may be useful. However, spatial locality is low, and >75% of reads are random.

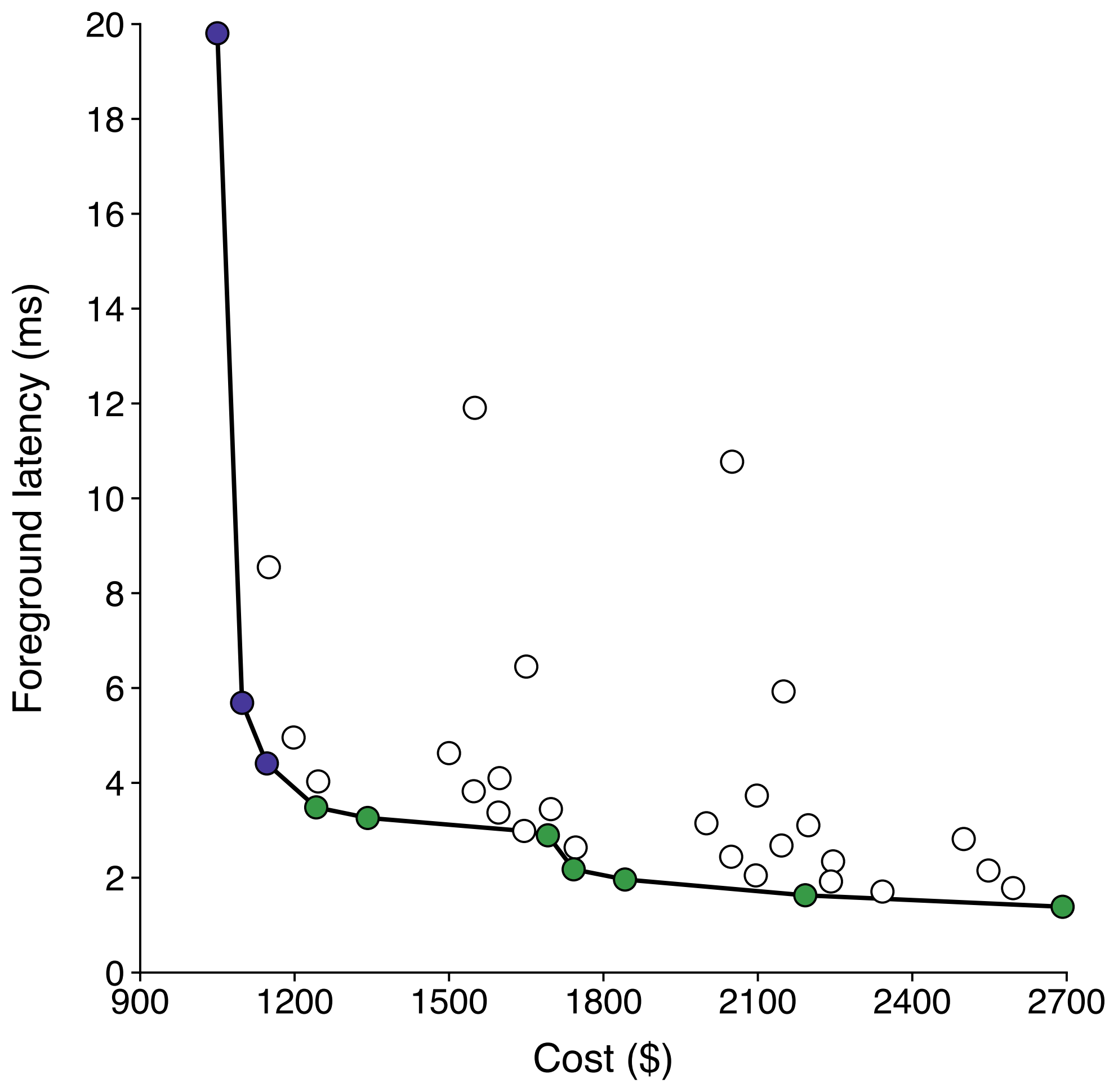
Summary: Messages represents a new HDFS workload, dominated by small files and random I/O. The dataset is very large and very cold.

Simulation

Q: Is adding a flash layer cost effective?

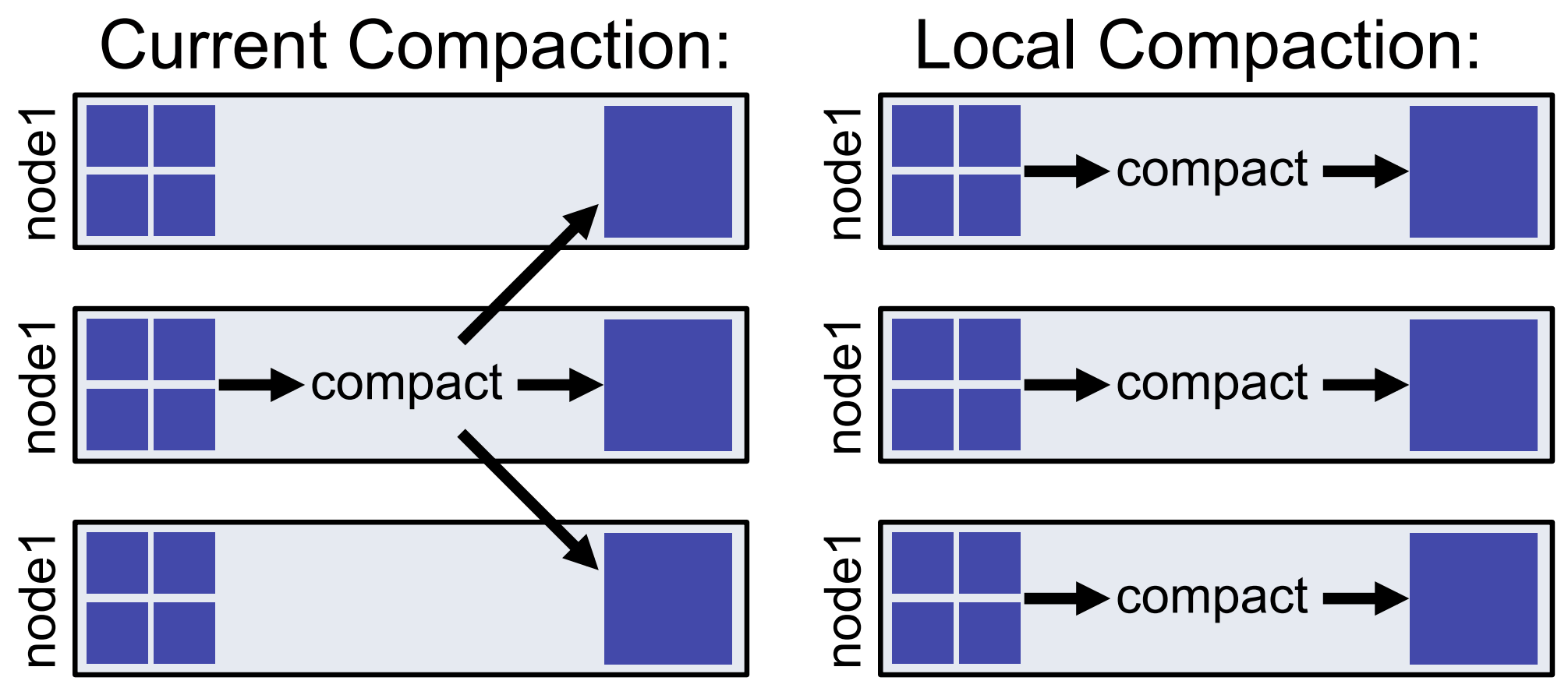
We compute monetary cost based on common hardware prices. We determine performance via simulation. We explore 36 systems (10, 15, 0r 20 disks, 10GB, 30GB, or 100GB of RAM, and 0, 60GB, 120GB, or 240GB of flash). Assumptions:

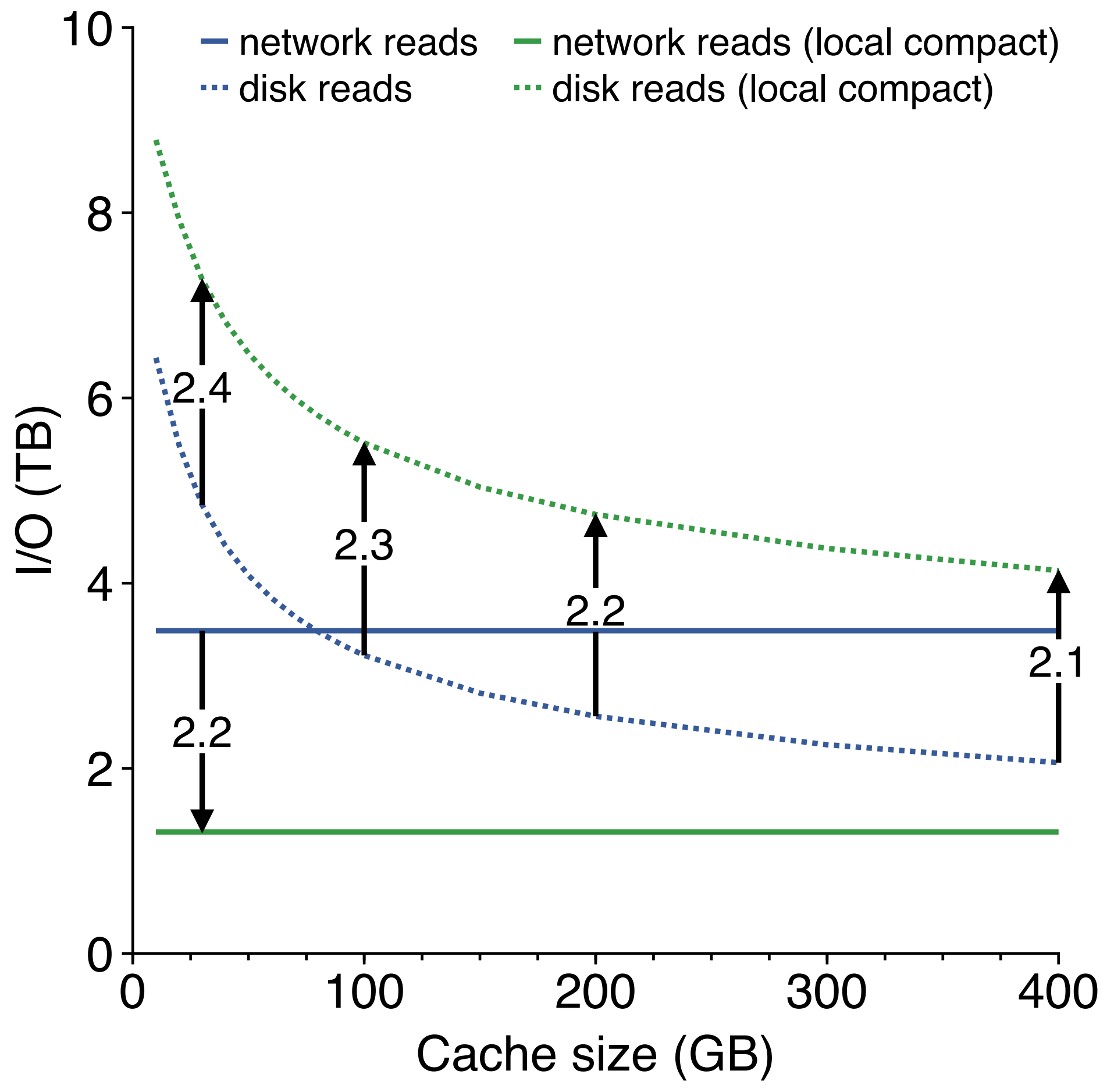
Hardware	Cost	Performance
HDD	$100/disk	10ms seek, 100MB/s
RAM	$5/GB	zero latency
Flash	$0.8/GB	0.5ms



A: Of the Pareto-optimal points, all but one have max flash (green) or min disk and RAM (blue)

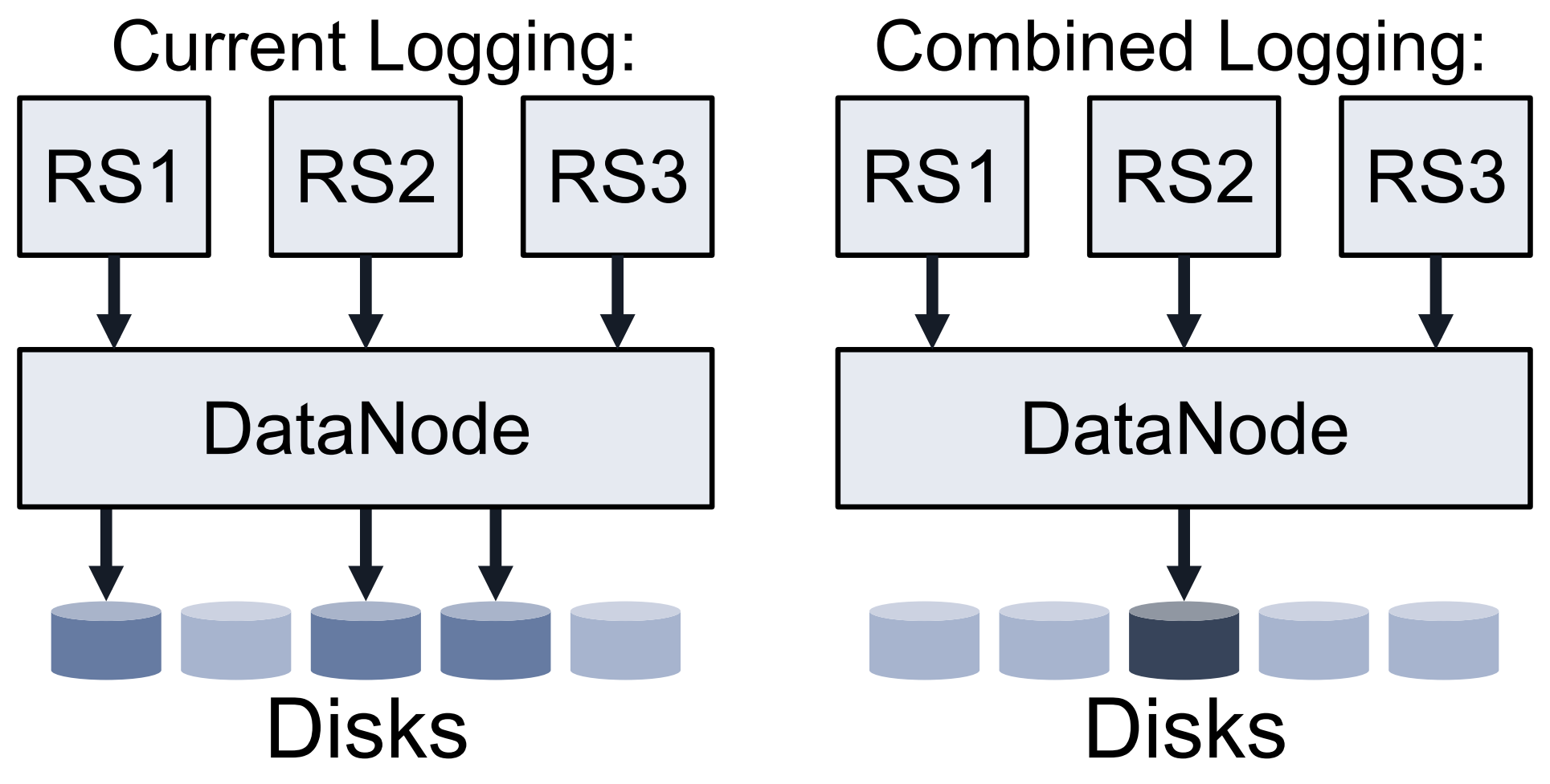
Q: Can support for compaction at the HDFS layer (i.e., local compaction) decrease network I/O?

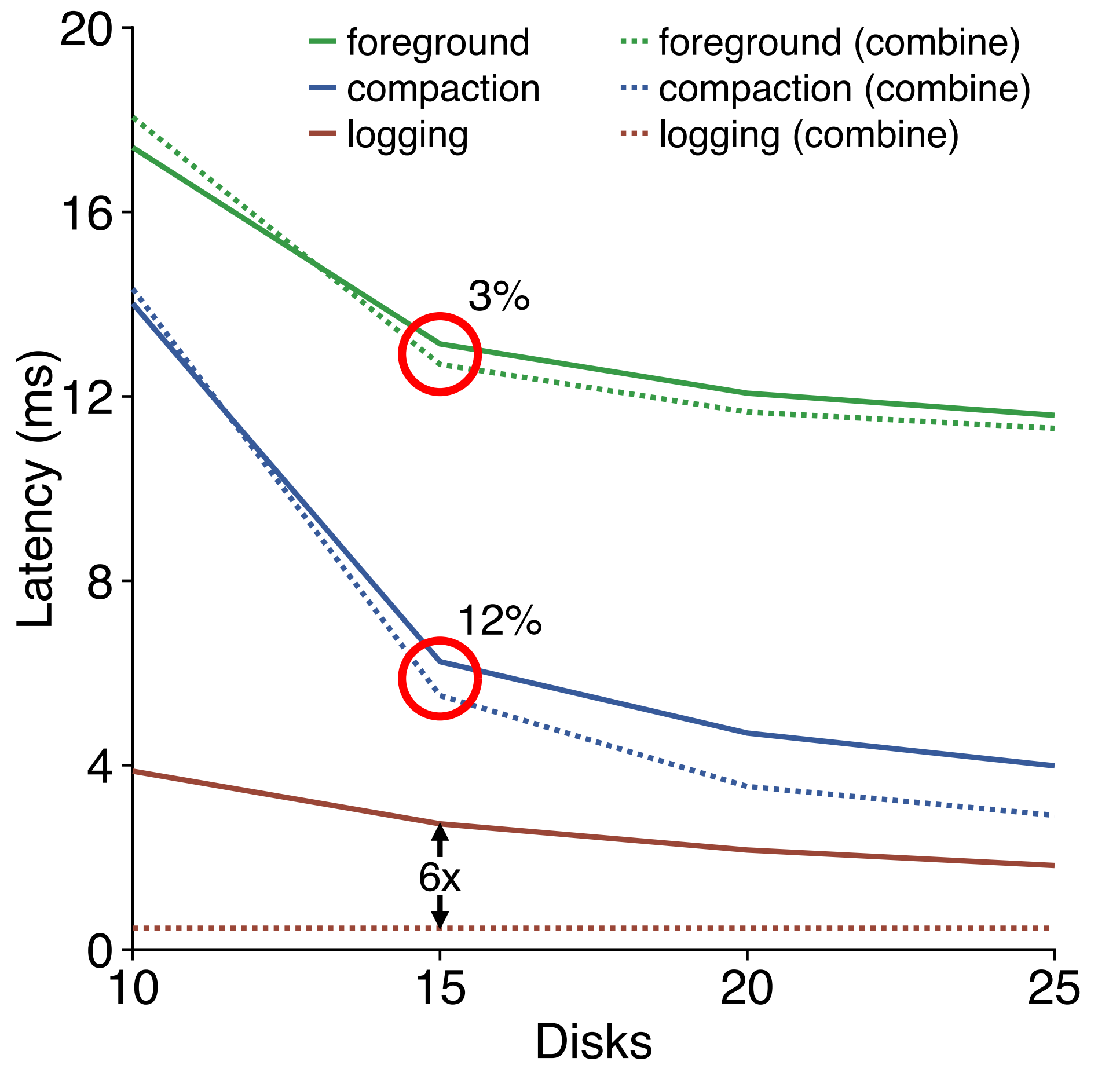




A: Local compaction converts 62% of network I/O (expensive) into disk I/O (cheaper)

Q: Can support for logging at the HDFS layer (i.e., combined logging) decrease disk seeks?





A: Combined logging makes log writes 6x faster without hurting other types of I/O

Conclusions

Relevance to important ideas

"*High sustained bandwidth is more important than low latency*" and "*multi-GB files are the common case.*"

~ GFS Paper

We find Messages is the opposite workload
- 50% of created files are <750KB
- >75% of reads are random

The use of layering "*proved to be vital for the verification and logical soundness*" of the THE operating system.

~ Dijkstra

We find layering is not free. Integration can
- Reduce network I/O by 62%
- Make log writes 6x faster

"*Tape is dead, disk is tape, flash is disk*"

~ Jim Gray

We find flash is not a suitable disk replacement
- Using pure flash would cost >$10K/machine
- However, a small SDD cache is a very cost-effective way to boost performance