Analysis of HDFS Under HBase
A Facebook Messages Case Study


Representative application
• Backend for texts, chats, and emails

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

Simulated stack
HDFS Traces
Local Traces (inferred)

Simulated stack
HDFS+HBase what-if s
Local Traces (inferred)

Local Storage
what-if s
Simulation Results

Intro: Why Messages?

Q: What is the read/write ratio across layers?
HDFS (excluding overheads): 1% writes
HDFS (including overheads): 21% writes
Local File System: 45% writes
Disk: 64% writes

Q: How much data is touched across layers?
HDFS (excluding overheads): 18% written
HDFS (including overheads): 77% written
Local File System: 91% written

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

Q: What is the read/write ratio across layers?
HDFS (excluding overheads): 1% writes
HDFS (including overheads): 21% writes
Local File System: 45% writes
Disk: 64% writes

Q: How much data is touched across layers?
HDFS (excluding overheads): 18% written
HDFS (including overheads): 77% written
Local File System: 91% written

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

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

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

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

A Facebook Messages Case Study

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

“We 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

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, or 20 disks, 100G, 300G, or 1000G of RAM, and 0, 60GB, 120GB, or 240GB of flash). Assumptions:

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

Targeted cost savings:
$10/GB flash

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

A: 20% are <128KB, 50% <750KB, 90% <6.3MB

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

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

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

Workload Analysis

Q: What is the read/write ratio across layers?
HDFS (excluding overheads): 1% writes
HDFS (including overheads): 21% writes
Local File System: 45% writes
Disk: 64% writes

Q: How much data is touched across layers?
HDFS (excluding overheads): 18% written
HDFS (including overheads): 77% written
Local File System: 91% written

Q: How large are created files?

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

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

Conclusions