

# Mastering Massive Data Volumes with Hypertable

*Doug Judd*

*[www.hypertable.org](http://www.hypertable.org)*



# Talk Outline

- Overview
- Architecture
- Performance Evaluation
- Case Studies

# Hypertable Overview

- Massively Scalable Database
- Modeled after Google's Bigtable
- Open Source (GPL 2 license)
- High Performance Implementation (C++)
- Thrift Interface for all popular High Level Languages: Java, Ruby, Python, PHP, etc
- Project started March 2007 @ Zvents

# Bigtable: the infrastructure that Google is built on

- Bigtable is a proven design
- Underpins 100+ Google services:  
YouTube, Blogger, Google Earth, Google Maps, Orkut, Gmail, Google Analytics, Google Book Search, Google Code, Crawl Database...
- Part of larger scalable computing stack:  
Bigtable + GFS + MapReduce
- Provides strong consistency

# Hypertable In Use Today



Baidu 百度



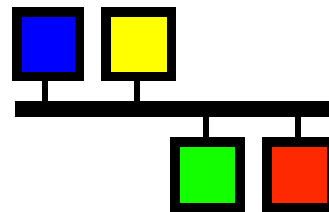
rediff.com



inepex



EPICS



ENDGAME SYSTEMS



Zvents  
Discover Things To Do



Tribalytic

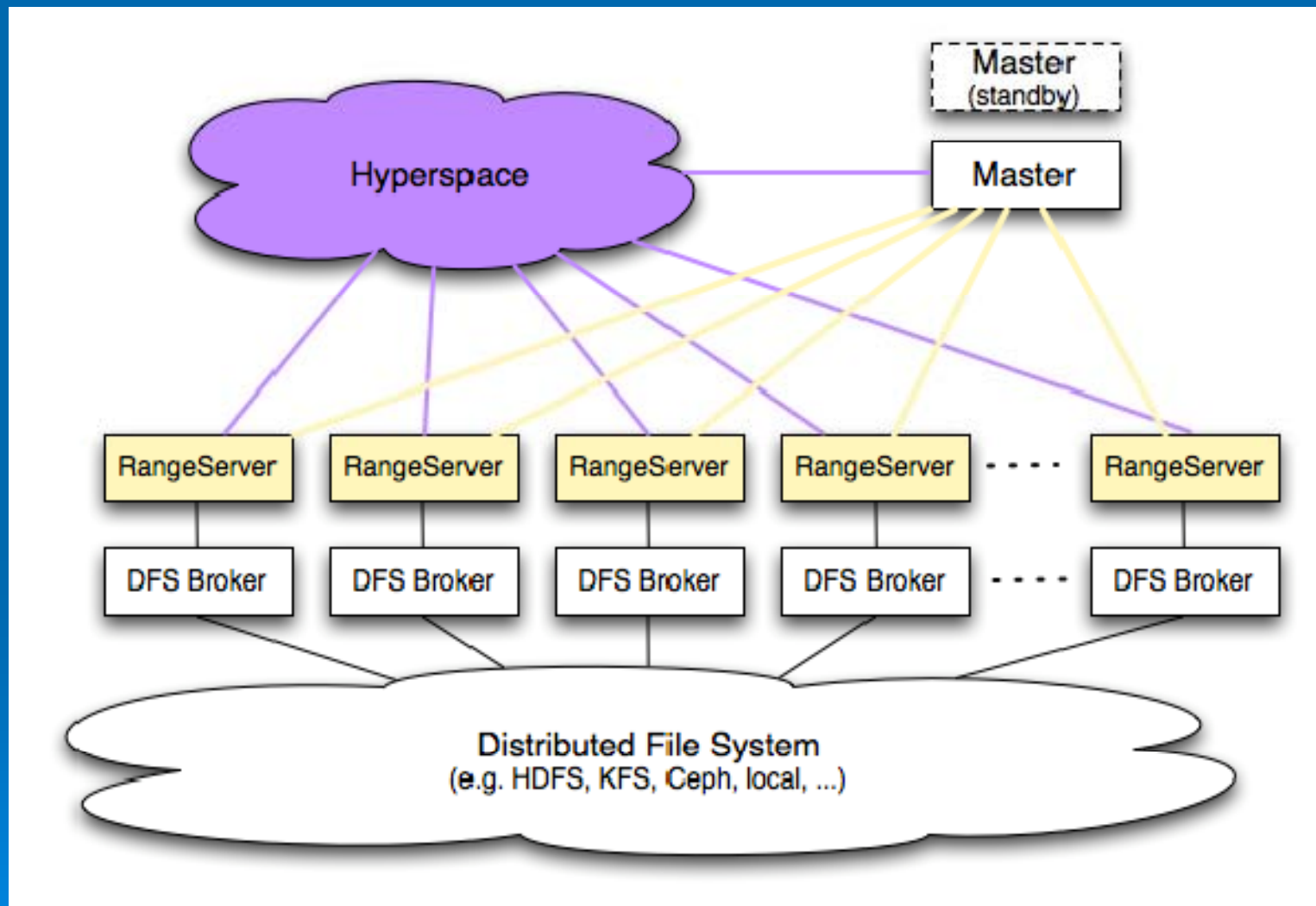


DEHEMS  
The Digital Environment Home Energy Management System

# Architecture



# System Overview



# Data Model

- Sparse, two-dimensional table with cell versions
- Cells are identified by a 4-part key
  - Row (string)
  - Column Family
  - Column Qualifier (string)
  - Timestamp

# Table: Visual Representation

**crawldb Table**

	<b>title</b>	<b>content</b>	<b>anchor</b>
<i>row key</i> →			<b>anchor:com.apple.www/</b>
com.facebook.www	Facebook   Home	<!DOCTYPE html ...	Facebook
			...
			<b>anchor:com.redherring.www/</b>
com.yahoo.www	Yahoo!	<html><head>...	Facebook
			<b>anchor:org.slashdot.www/</b>
com.zvents.www	Discover Things To Do - Zvents	<html xmlns="http...	Zvents
org.hypertable.www	Hypertable: An Open Source, High Performance, ...	<!DOCTYPE html ...	

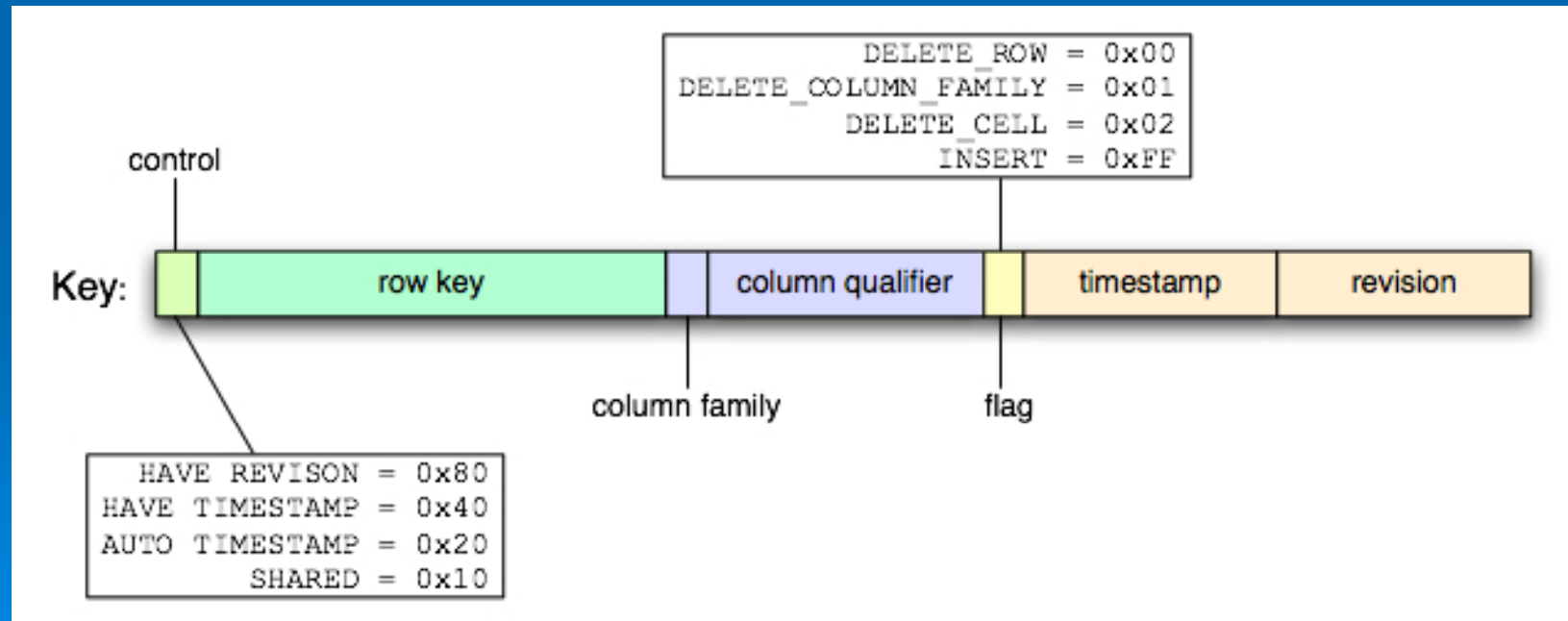
# Table: Actual Representation

## crawldb Table

key	value
com.facebook.www title 2008-02-11 15:14:01	Facebook   Home
com.facebook.www title 2008-02-03 19:27:57	Facebook   Home
com.facebook.www title 2008-01-22 08:46:28	Facebook   Home
com.facebook.www content 2008-02-11 15:14:01	<!DOCTYPE html PUBLIC "-//W3C//DTD...
com.facebook.www content 2008-02-03 19:27:57	<!DOCTYPE html PUBLIC "-//W3C//DTD...
com.facebook.www content 2008-01-22 08:46:28	<!DOCTYPE html PUBLIC "-//W3C//DTD...
com.facebook.www anchor:com.apple.www/ 2008-02-11 15:14:01	Facebook
com.facebook.www anchor:com.apple.www/ 2008-02-03 19:27:57	Facebook
com.facebook.www anchor:com.apple.www/ 2008-01-22 08:46:28	Facebook
com.facebook.www anchor:com.redherring.www/ 2008-02-11 15:14:01	Facebook
com.facebook.www anchor:com.redherring.www/ 2008-02-03 19:27:57	Facebook
com.yahoo.www title 2008-02-10 21:12:09	Yahoo!
com.yahoo.www title 2008-02-04 03:46:22	Yahoo!
com.yahoo.www title 2008-01-22 08:46:28	Yahoo!
com.yahoo.www content 2008-02-10 21:12:09	<html><head><meta http-equiv="Content-...
com.yahoo.www content 2008-02-04 03:46:22	<html><head><meta http-equiv="Content-...
...	...

# Anatomy of a Key

- Column Family is represented with 1 byte
- Timestamp and revision are stored big-endian ones-compliment
- Simple byte-wise comparison



# Scaling (part I)

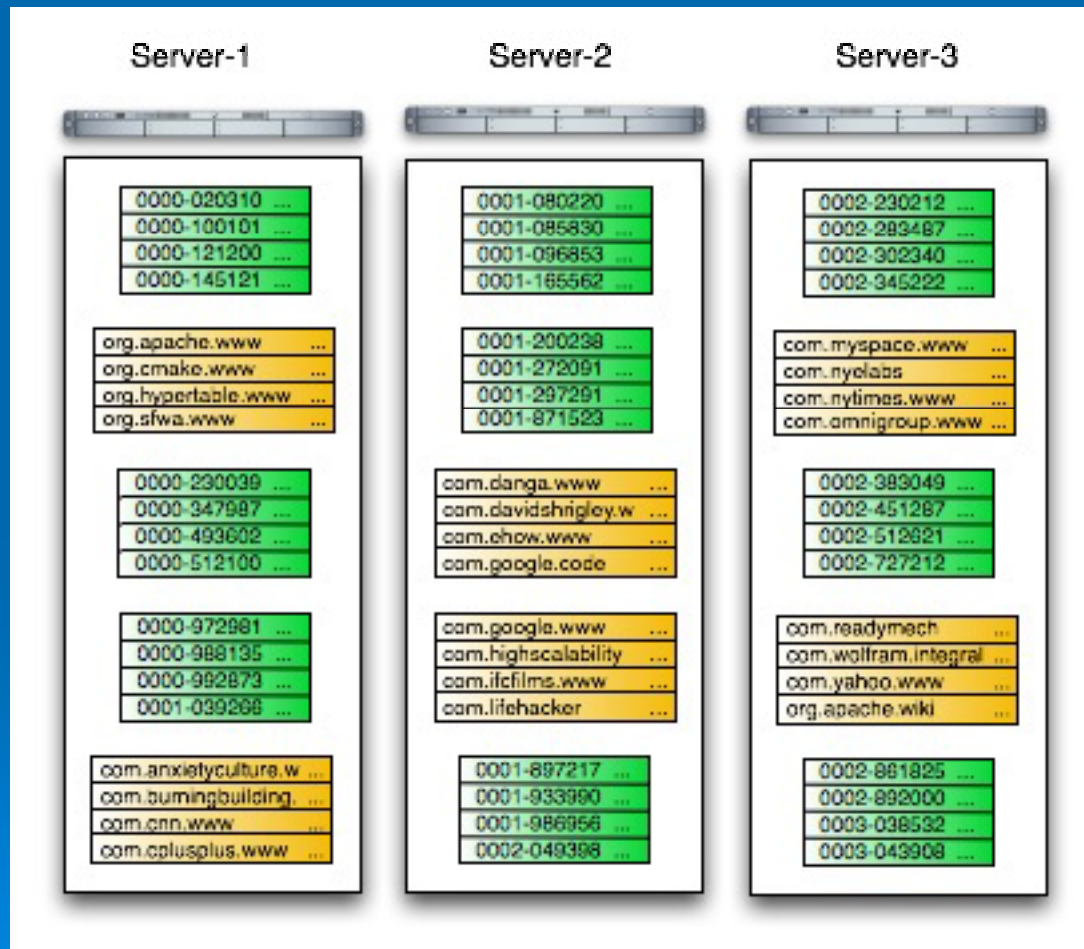
session table

0000-020310 ...
0000-100101 ...
0000-121200 ...
0000-145121 ...
0000-230039 ...
0000-347987 ...
0000-493602 ...
0000-512100 ...
0000-972981 ...
0000-988135 ...
0000-992873 ...
0001-039266 ...
0001-080220 ...
0001-085830 ...
0001-096853 ...
0001-165562 ...
0001-200238 ...
0001-272091 ...
0001-297291 ...
0001-871523 ...
0001-897217 ...
0001-933990 ...
0001-986956 ...
0002-049398 ...
0002-230212 ...
0002-283487 ...
0002-302340 ...
0002-345222 ...
0002-383049 ...
0002-451287 ...
0002-512621 ...
0002-727212 ...
...

crawldb table

com.anxietyculture.com ...
com.burningbuilding.www ...
com.cnn.www ...
com.cplusplus.www ...
com.danga.www ...
com.davidshrigley.www ...
com.ehow.www ...
com.google.code ...
com.google.www ...
com.highscalability ...
com.ifcfilms.www ...
com.lifehacker ...
com.myspace.www ...
com.nyelabs ...
com.nytimes.www ...
com.omnigroup.www ...
com.readymech ...
com.wolfram.integrals ...
com.yahoo.www ...
org.apache.wiki ...
org.apache.www ...
org.cmake.www ...
org.hypertable.www ...
org.sfga.www ...

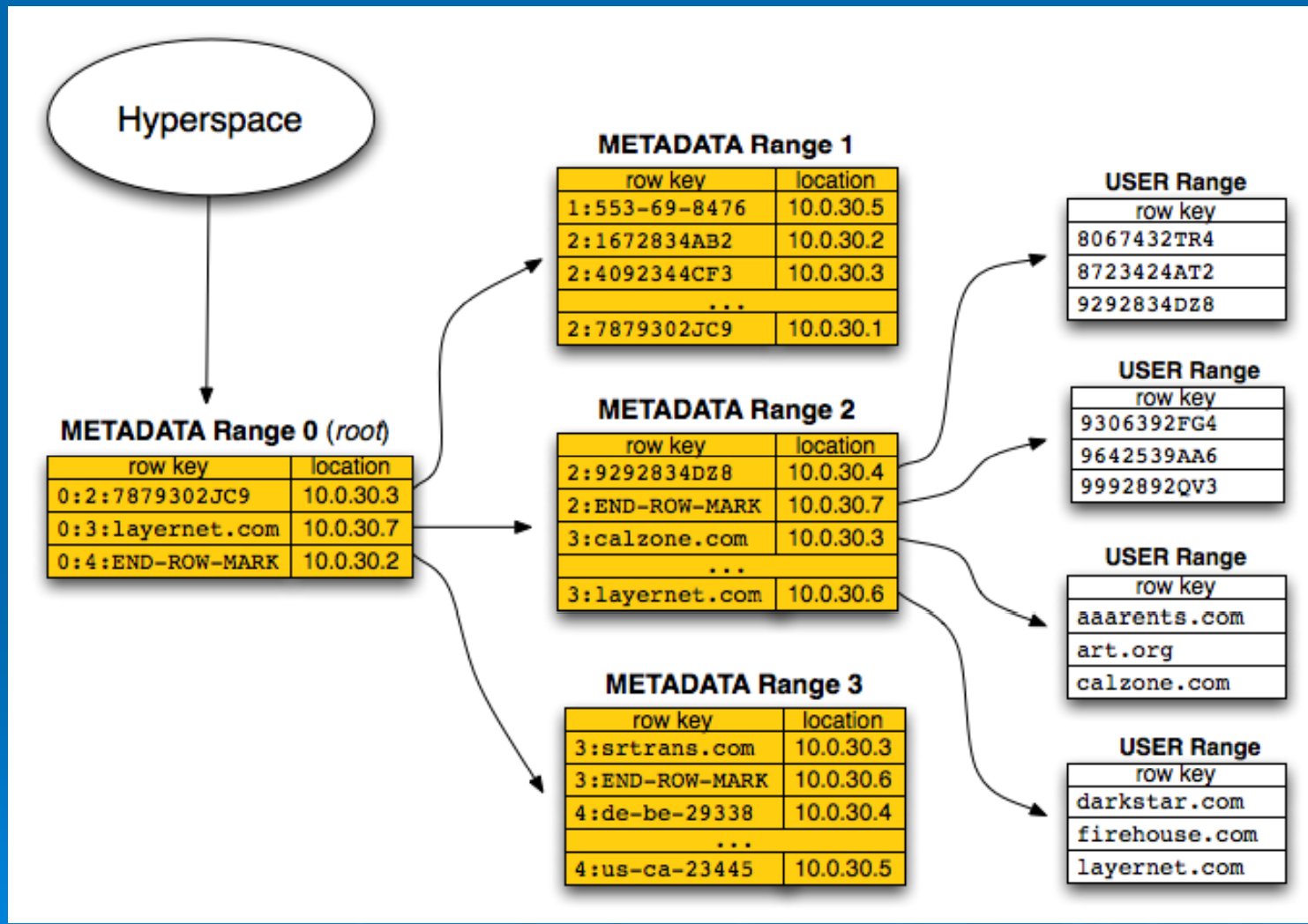
# Scaling (part II)



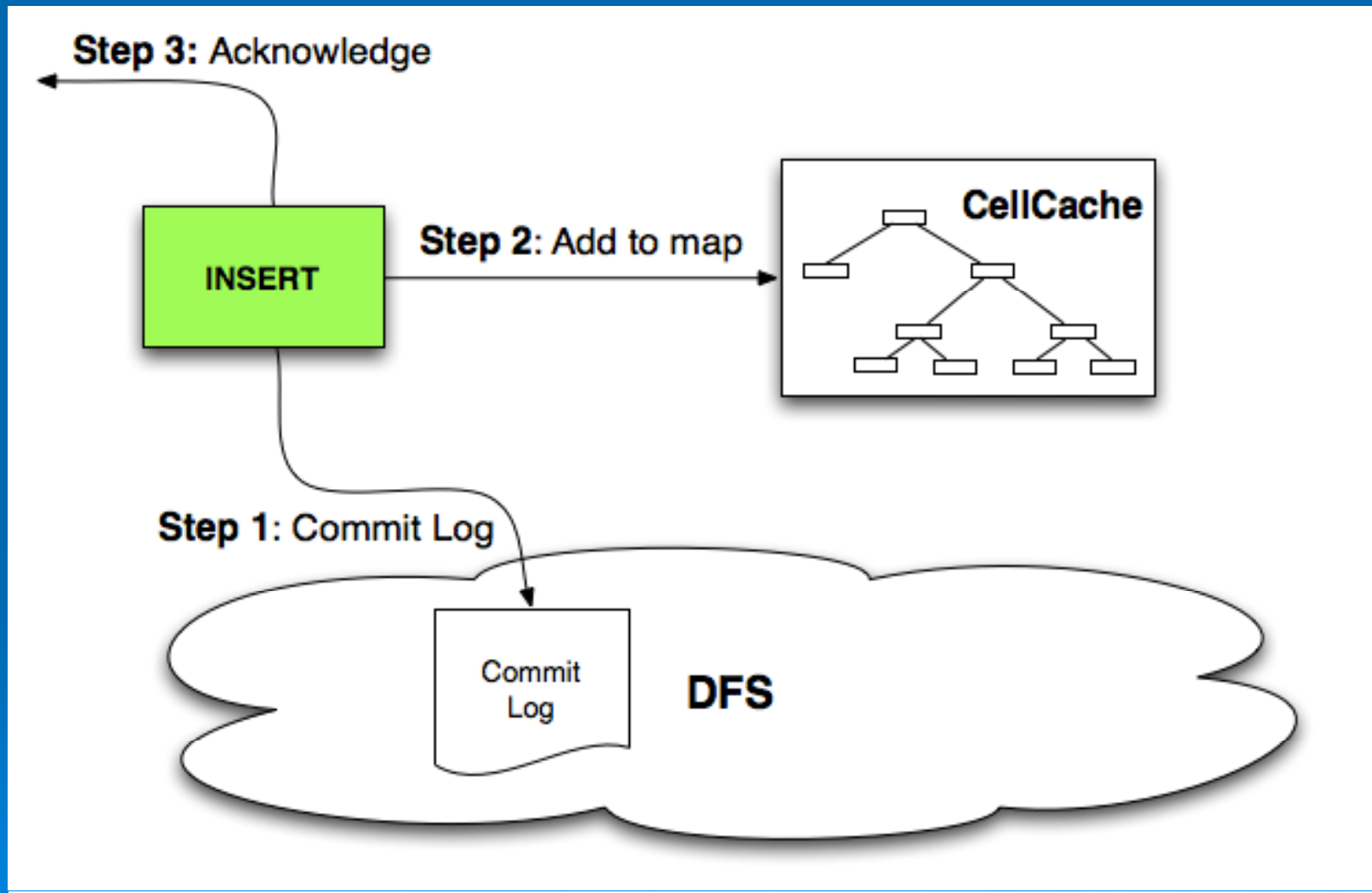
# Scaling (part III)



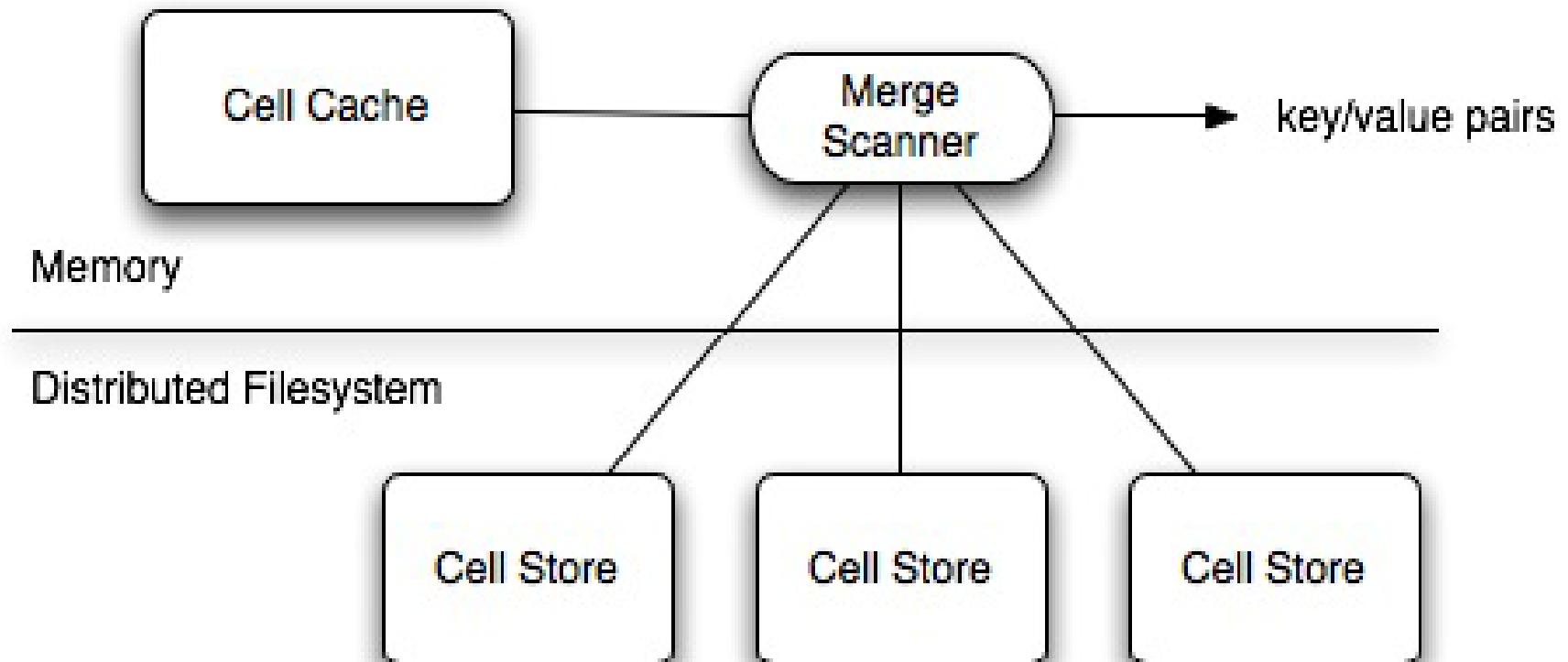
# Request Routing



# Insert Handling

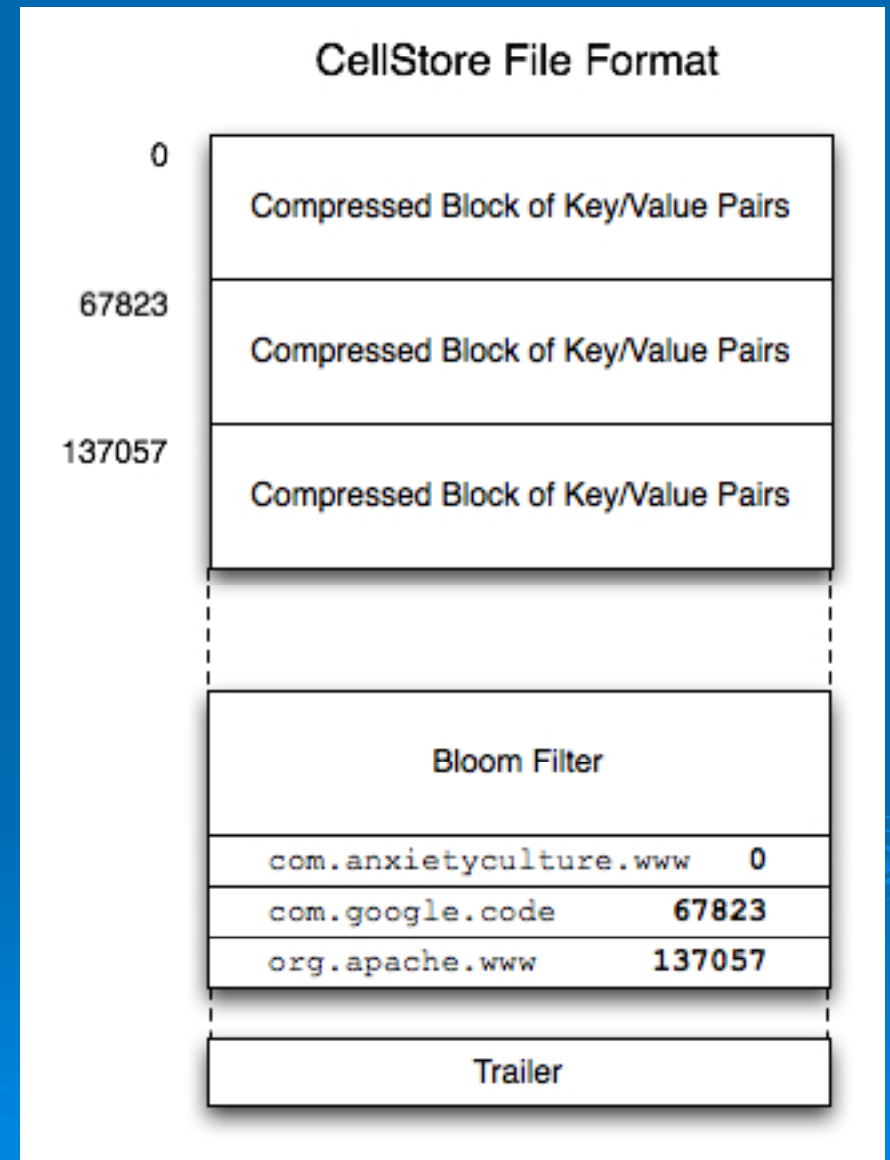


# Query Handling



# CellStore Format

- Immutable file containing sorted sequence of key/value pairs
- Sequence of 65K blocks of compressed key/value pairs



# Compression

- Cell Store blocks are compressed
- Commit Log updates are compressed
- Supported Compression Schemes
  - zlib (--best and --fast)
  - lzo
  - quicklz
  - bmz
  - none

# Caching

## ➤ Block Cache

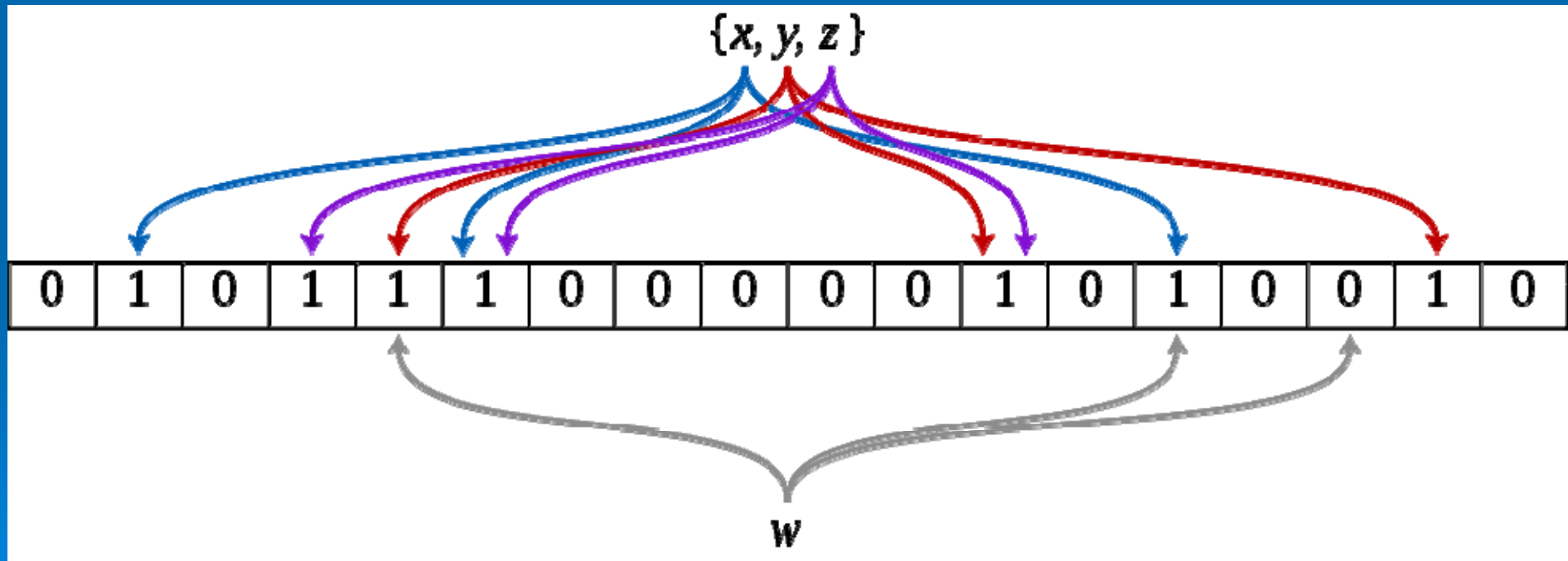
- Caches CellStore blocks
- Blocks are cached uncompressed
- Dynamically adjusted size based on workload

## ➤ Query Cache

- Caches query results

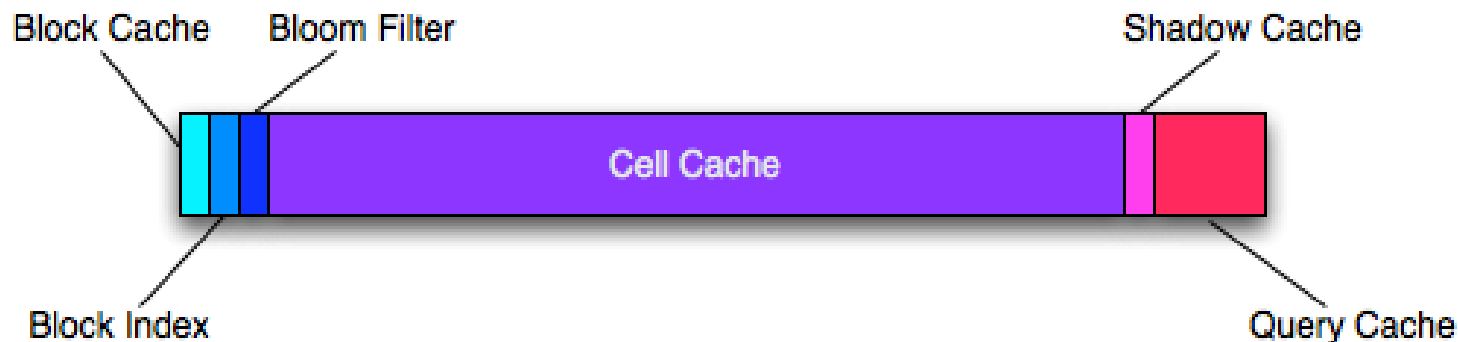
# Bloom Filter

- Probabilistic data structure associated with every CellStore
- Tells you if key is definitively **not** present

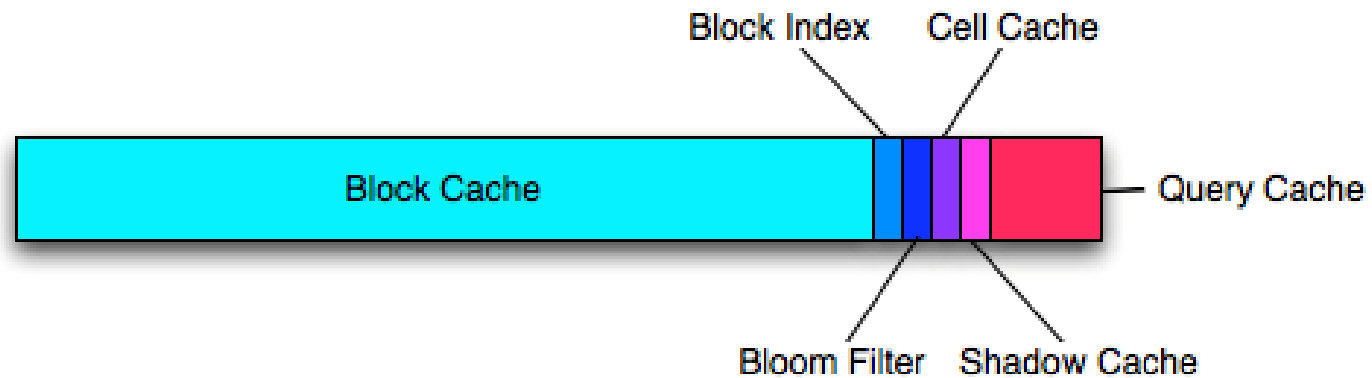


# Dynamic Memory Allocation

## Write Heavy Workload



## Read Heavy Workload



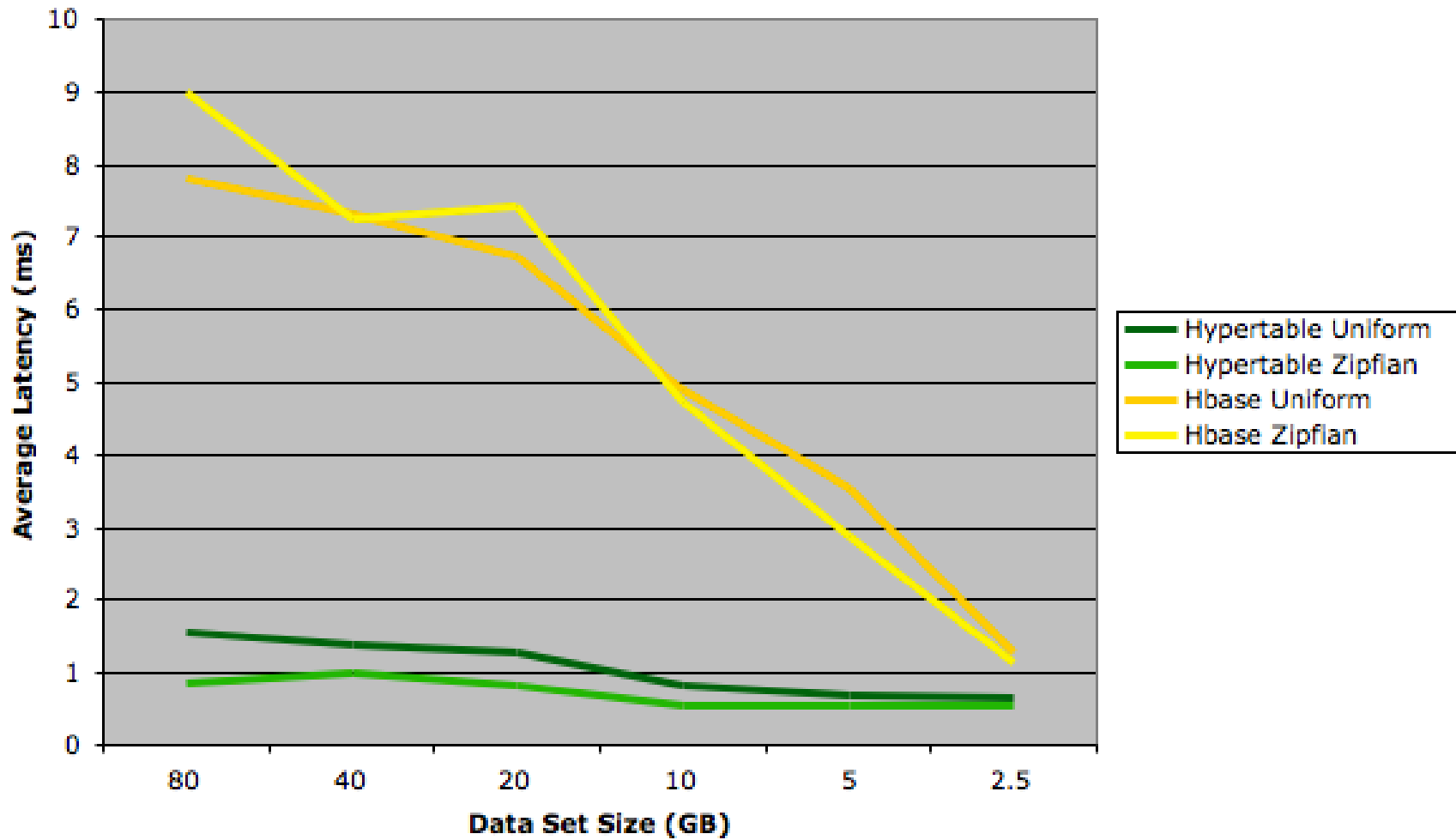
# Performance Evaluation



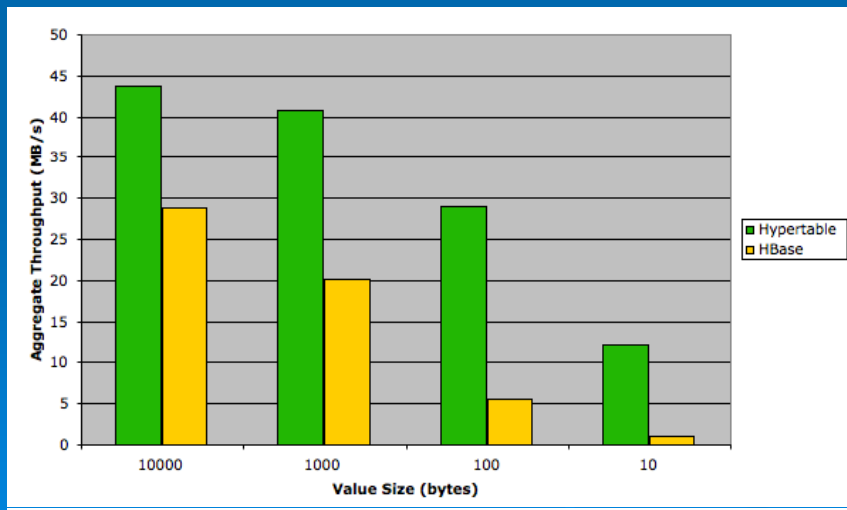
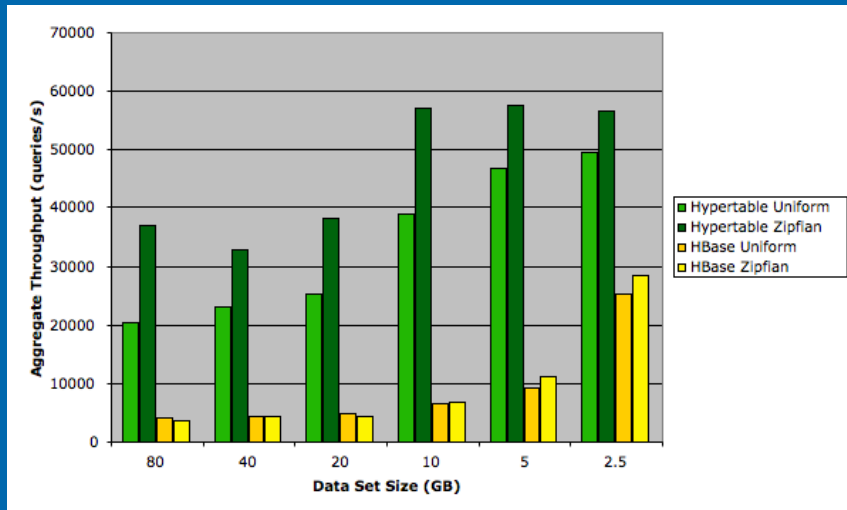
# Setup

- Modeled after Test described in Bigtable paper
- 1 Test Dispatcher, 4 Test Clients, 4 Tablet Servers
- Test was written entirely in Java
- Hardware
  - 1 X 1.8 GHz Dual-core Opteron
  - 10 GB RAM
  - 3X 250GB SATA drives
- Software
  - HDFS 0.20.2 running on all 10 nodes, 3X replication
  - HBase 0.20.4
  - Hypertable 0.9.3.3

# Latency



# Throughput



Test	Hypertable Advantage Relative to HBase (%)
Random Read Zipfian 80 GB	925
Random Read Zipfian 20 GB	777
Random Read Zipfian 2.5 GB	100
Random Write 10KB values	51
Random Write 1KB values	102
Random Write 100 byte values	427
Random Write 10 byte values	931
Sequential Read 10KB values	1060
Sequential Read 1KB values	68
Sequential Read 100 byte values	129
Scan 10KB values	2
Scan 1KB values	58
Scan 100 byte values	75
Scan 10 byte values	220

# Detailed Report

<http://blog.hypertable.com/?p=14>

# Case Studies



# Zvents

- Longest running Hypertable deployment
- Analytics Platform - Reporting
  - Per-listing page view graphs
  - Change log
- Ruby on Rails - HyperRecord

# Rediff.com

- rediffmail SPAM classification
- Several hundred front-end web servers
- tens of millions requests/day
- Peak rate 500 queries/s
- 99.99% sub-second response time

# Best Fit Use Cases

- Serve massive data sets to live applications
- It's ordered - good for applications that need to scan over data ranges (e.g. time series data)
- Typical workflow:  
logs -> HDFS -> MapReduce -> Hypertable

# Resources

- Project site:  
[www.hypertable.org](http://www.hypertable.org)
- Twitter: [hypertable](https://twitter.com/hypertable)
- Commercial Support:  
[www.hypertable.com](http://www.hypertable.com)

# Q&A

