HBase: The Hadoop Database

February 21, 2012
Mark Stetzer
@stetzer

Who is this guy?

- Director of Engineering @ ChaCha
- Founder of IndyHUG
- Implemented Hadoop @ ChaCha
- Using Hadoop regularly since 2009
- Using HBase regularly since 2011

What’s the plan?

- What is HBase?
- Why (not) use HBase?
- Architecture overview
- Client API overview
- Tenants of schema design
- Demo
- Additional features

HBase is...

- Open source
- Distributed
- Column-oriented
- Fault-tolerant
- Linearly-scalable
- Capable of managing 1 petabyte (or more?) of data
- Built for realtime random access to data

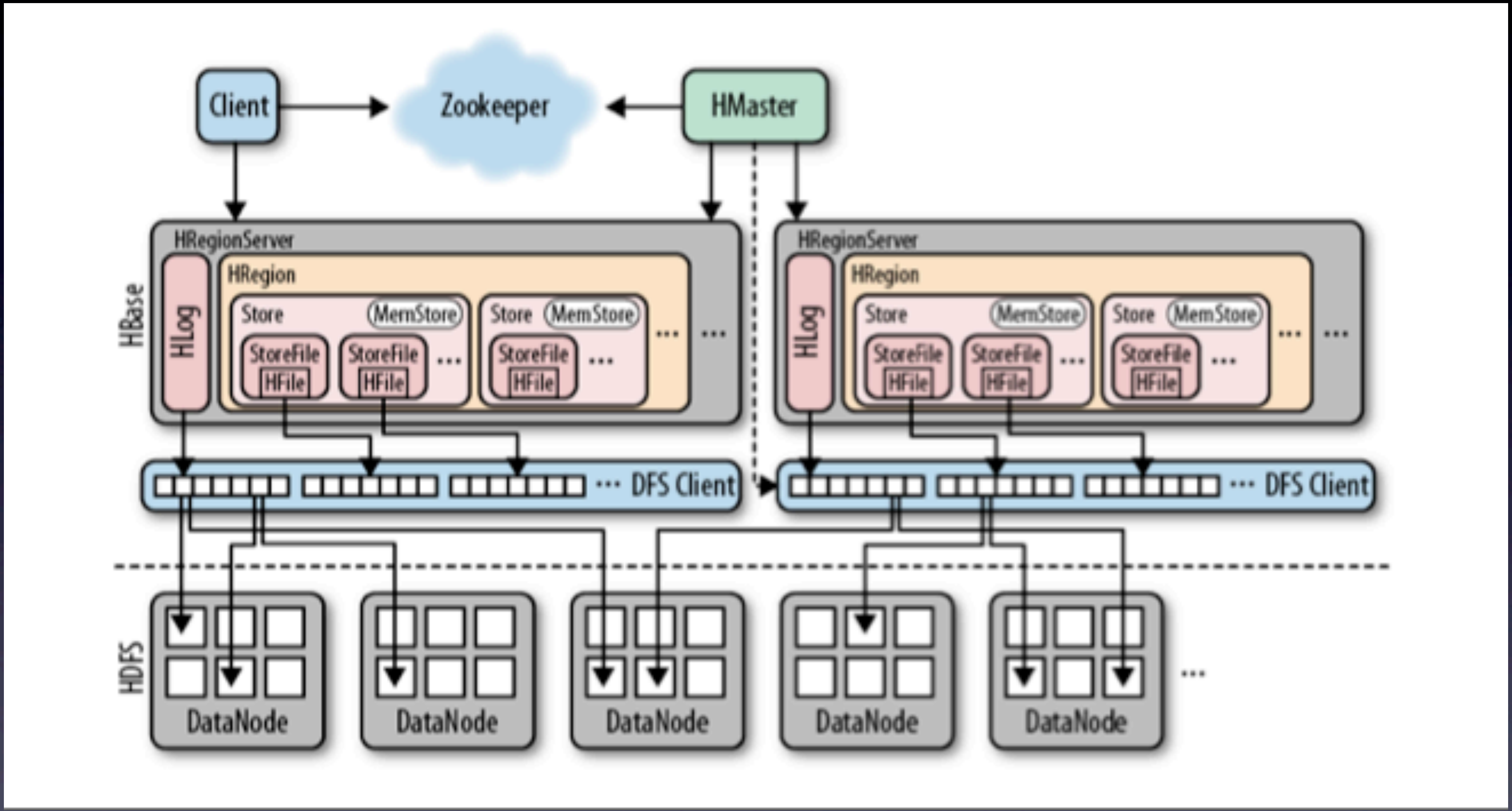
Why use HBase?

- If you have a large amount of data (100+ million rows)...
- ...that you need realtime access to...
- ...and you can live without traditional RDBMS tools...
- ...then HBase could be worth investigating.

Good use case to deploy?

- Usage of RDBMS is becoming troublesome
- Master/slave isn't working
- Sharding is hard
- Growing linearly with commodity hardware is more affordable than buying bigger & bigger

Architecture



© HBase The Definitive Guide by Lars George (Oreilly)

Architecture

- Multiple masters, many region servers (in charge of handling data requests)
- Write to HDFS; S3 and other filesystems also supported
- Zookeeper used to coordinate locks, store metadata, etc.
- Write-ahead log (WAL) used for data consistency

Architecture

- Data written to memstore, eventually flushed to disk
- Regions split when too big; handling passed off to multiple region servers
- Periodic compaction; somewhat similar to defrag
- Automatic compression of tables: GZIP, Snappy, LZO

Data structure (sorted map)

Row key	Column key	Timestamp	Cell
pepsi-20120214	daily:questions	1.3298E+12	2013
pepsi-20120215	daily:questions	1.3298E+12	1987
pepsi-20120215	daily:questions	1.3298E+12	1892

Note two values for same row (diff times)

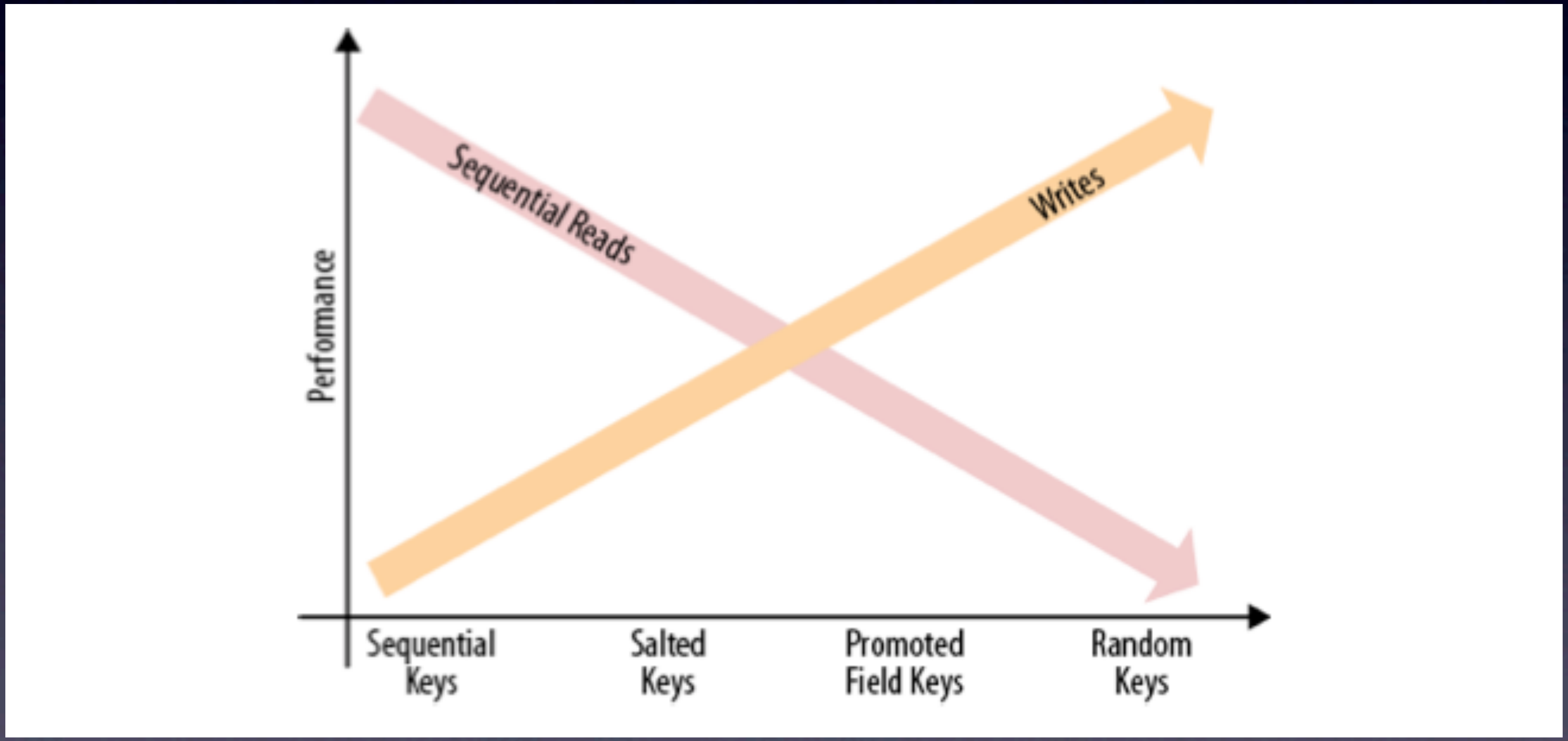
Client API

- Working with arbitrary byte arrays (no real size constraints)
- get(row)
- put(row, map<col, val>)
- scan(key range, filter)
- increment(row, columns)
- delete(row, columns)
- Others (batch ops, checkAndPut, metadata access, M/R)

Schema Design

- All about the key
- Data is stored sorted by row key
- If storing time-sequential data, prefixing key with timestamp bad (causes hot regions)
- Common workarounds: promoted fields, salted keys, completely random keys

Sequential Read vs Write Performance



© HBase The Definitive Guide by Lars George (Oreilly)

Tall-narrow vs flat-wide tables

- Tall-narrow: few columns, many rows
- Flat-wide: few rows, many columns
- Since HBase splits on row keys, tall-narrow usually best choice
- Promote fields to key for efficient query performance using partial key scans

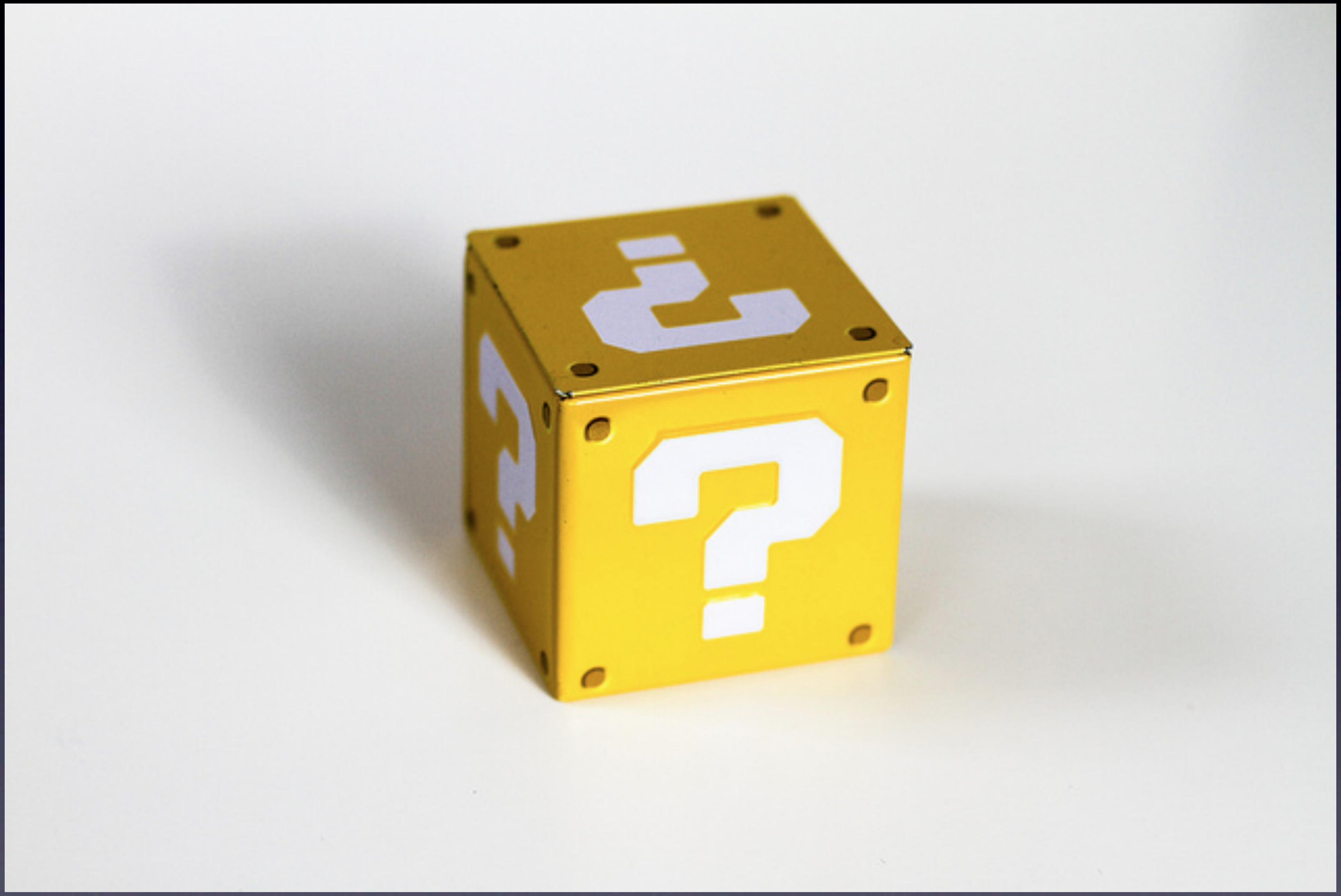
Demo

- HBase and required services
- Interactive console: “hbase shell”
- Loading data
- Scanning data

Additional features

- Map/Reduce integration
- Secondary indexes
- Coprocessors (0.92+)
- Bloom filters
- Versioning
- TTLs

Questions?



© Flickr user [F]oxymoron