HBasics: *An Introduction to Hadoop HBase*

ApacheCon, Amsterdam, 2009

Michael Stack

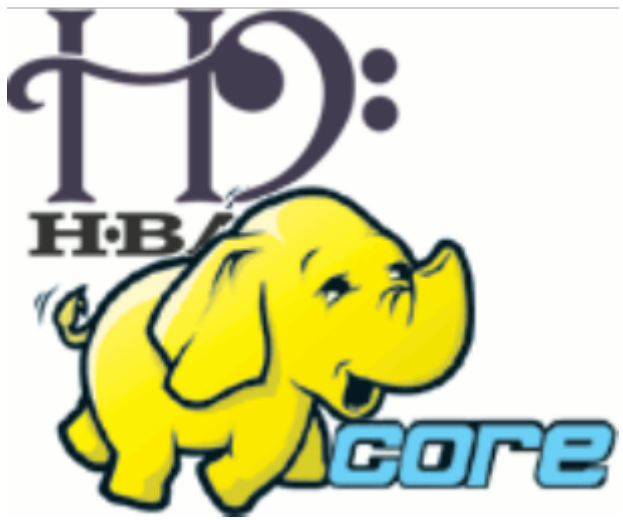
Powerset, a Microsoft Company

Overview

1. What is HBase
2. Data Model
3. Implementation
4. Using HBase
5. Project Status
6. Powered-by

What is HBase?

- Distributed database modeled on Google's Bigtable
  - Bigtable: A Distributed Storage System for Structured Data by *Chang et al.*
- Structures data as tables of *column-oriented* rows
- Online access to Big Data
  - Random-access and updates immediately available
    - Whereas MapReduce is batch-processing
  - Multiple processes can update individual records asynchronously
- Project Goal: Scalable data store
  - Billions of rows X millions of columns X thousands of versions
  - Petabytes across thousands of "commodity" servers
- Apache Hadoop subproject
  - Runs on top of Hadoop HDFS
  - Data is replicated, it scales, etc



What HBase is not...

A SQL Database!

- Not relational
- No joins
- No sophisticated query engine
- No column typing
- No SQL, no ODBC/JDBC, Crystal Reports, etc.
- No transactions
- No secondary indices

Not a drop-in replacement for your RDBMS

Whats HBase for?

- Traditional RDBMS bounded by size of single node
  - Ergo: Big Iron
- Scaling beyond is generally an appendage
  - Partitioning/replication complicated, expensive
  - Comes at some compromise in RDBMS facility
    - Denormalize
    - No database-level joins, etc.
- Whereas, HBase, is a scalable data store
  - Built to scale from the get-go
  - "Commodity" HW rather than the exotic
- HBase replacement for 'scaled' RDBMS

More on whats' HBase for?

- HBase has limited feature-set
  - CRUD: Create, Read, Update, & Delete
  - Primary-key access
  - No joins in db -- application-level (MapReduce)
  - *How much does this differ from "scaled" RDBMS?*
- Canonical use case is the *Webtable*
  - Table of web crawls keyed by URL, crawldate denotes version
  - Columns of webpage, parse, page attributes, etc.
  - Webtable fields batch (MapReduce) and random access
    - Batch: analytics and derivations
    - Random access: crawler updates, serving cached pages

Data Model: 1 of 2

- Labeled tables of rows X columns X timestamp
  - Map where cells addressed by row/column/timestamp
  - As (perverse) java declaration:

```
SortedMap<byte [], SortedMap<byte [],
  List<Cell>>>> hbase =
    new TreeMap<ditto>(new RawByteComparator());
```

- Row keys uninterpreted byte arrays: E.g. an URL
  - Rows are ordered by Comparator (Default: byte-order)
  - Row updates are atomic; even if hundreds of columns
- Columns grouped into *columnfamilies*
  - Columns have columnfamily *prefix* and then *qualifier*
    - E.g. *webpage:mimetype, webpage:language*
    - *Columnfamily* 'printable', *qualifier* arbitrary bytes
  - Columnfamilies in table schema but qualifiers, willy-nilly
  - Tables are 'sparse'; not all columns populated in a row

Data Model: 2 of 2

- Cell is uninterpreted byte [] and a timestamp
  - E.g. webpage content
- Tables partitioned into Regions
  - Region defined by *start* & *end* row
  - Regions are the 'atoms' of distribution
    - Deployed around the cluster

Implementation: 1 of 3

- Master, one or more RegionServers, & Clients
  - Single-master cluster
  - Like HDFS & MR in Hadoop
  - Own start/stop cluster scripts
  - Own hbase-*.xml config.

```
durruti:trunk stack$ ./bin/hbase
Usage: hbase <command>
where <command> is one of:
  shell            run the HBase shell
  master           run an HBase HMaster node
  regionserver     run an HBase HRegionServer node
  rest             run an HBase REST server
  thrift           run an HBase Thrift server
  zookeeper        run a Zookeeper server
  migrate          upgrade an hbase.rootdir
 or
  CLASSNAME        run the class named CLASSNAME
Most commands print help when invoked w/o paramete
```

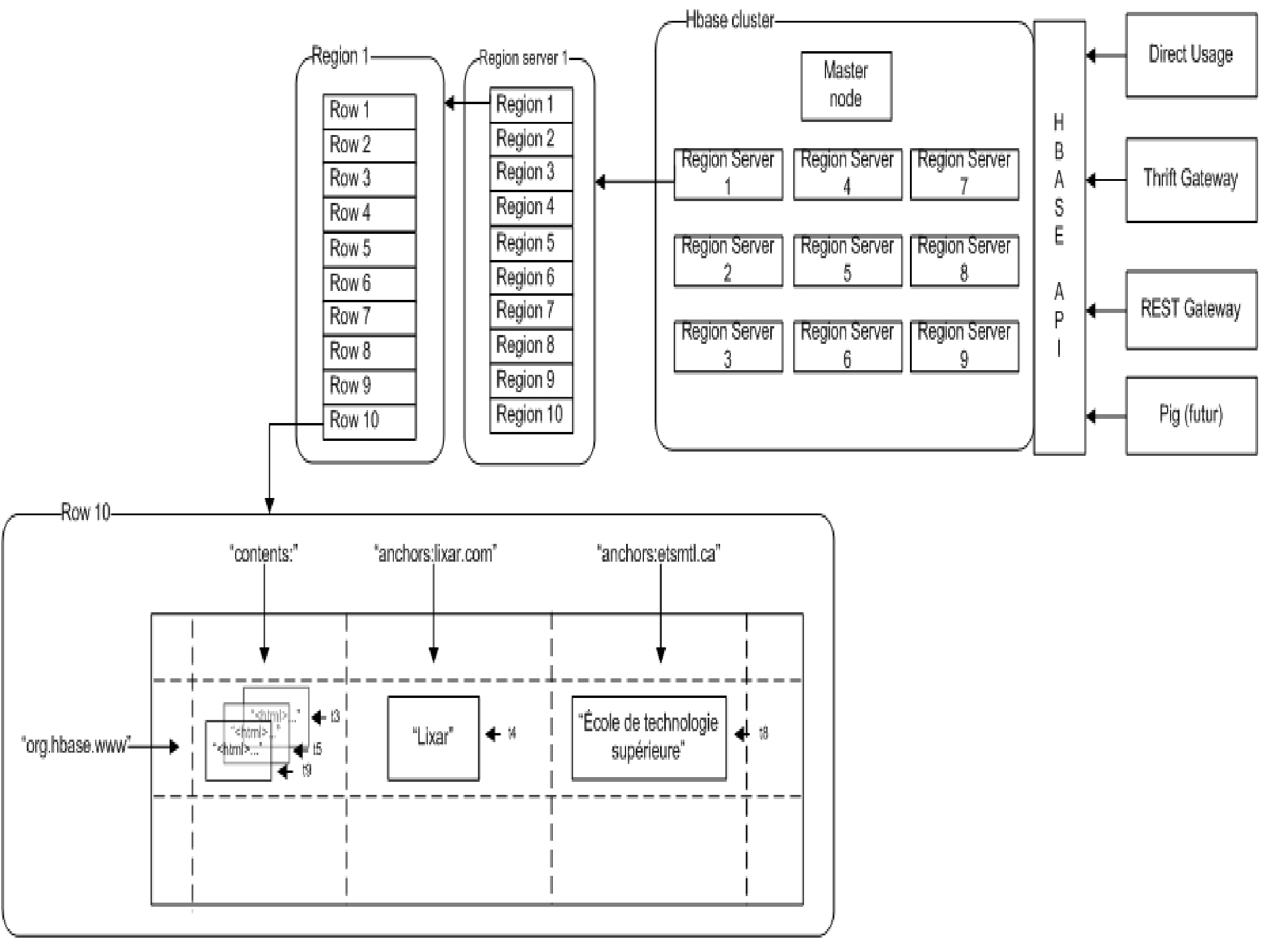
- Cluster carries 0->N labeled Tables
- Master assigns Table Regions to RegionServers
  - Master manages schema edits
  - Reallocation of regions when crash
  - Lightly loaded
- RegionServer carries 0->N regions
  - RegionServer keeps *commit log* of every update
  - Used recovering lost regionserver edits
    - More on this later...
  - Edits go first to RS commit log, then to Region

Implementation: 2 of 3

- Each Region is made of Stores
  - *Columnfamily* from data model implemented as a Store
    - All in columnfamily stored together; i.e. *CF-orientated*
      - Wide tables OK since only pertinent CF participate
      - Good for sparse data, only data stored, no need of a NULL representation
      - CF members should have similar character/access
  - Store MemCache takes on Region writes: flushes when full
  - Flush adds a StoreFile
  - StoreFile key is r/c/t and value is cell
  - Per Store, when > N StoreFiles, compacted in background
- When Regions get too big, they are split
  - RegionServer manages split
    - Fast, no data rewrite
    - Daughters server top and bottom halves of parent
  - Master is informed, parent is offlined, new daughters deployed

Implementation: 3 of 3

- Traffic flow
  - Client ***initially*** goes to Master for region hosting row
  - Master supplies client a Region specification and host
  - Client caches; goes direct to RegionServer thereafter
  - If fault (split/crash), returns to Master to freshen its cache
  - Region locations in *catalog* tables
- HBase is made of Hadoop Parts
  - Customized Hadoop RPC
  - MapFile for HStoreFiles
  - SequenceFile for commit logs, etc


Hbase cluster
Master node
Region Server 1
Region Server 4
Region Server 7
Region Server 2
Region Server 5
Region Server 8
Region Server 3
Region Server 6
Region Server 9
H B A S E A P I
Direct Usage
Thrift Gateway
REST Gateway
Pig (futur)
Region 1
Row 1
Row 2
Row 3
Row 4
Row 5
Row 6
Row 7
Row 8
Row 9
Row 10
Region server 1
Region 1
Region 2
Region 3
Region 4
Region 5
Region 6
Region 7
Region 8
Region 9
Region 10
Row 10
"contents:"
"anchors:lixar.com"
"anchors:etsmtl.ca"
"org.hbase.www"
"<html>..."
"<html>..."
"<html>..."
t3
t5
t9
"Lixar"
t4
"École de technologie supérieure"
t8

Connecting to HBase

Java client

```
HBaseConfiguration config = new HBaseConfiguration();
HTable table = new HTable(config, "myTable");
Cell cell = table.get("myRow",
  "myColumnFamily:columnQualifier1");
```

Non-Java clients

- Thrift server hosting HBase client instance
  - Sample ruby, c++, c#!, & java (via thrift) clients
- REST server hosts HBase client
  - JSON or XML

MapReduce

- TableInput/OutputFormat
  - HBase as MapReduce source or sink
- Cascading HBaseTap
- PIG connector

HBase Shell

- JRuby IRB
  - "real" shell
- Non-SQL (intentional) “DSL”
  - get
  - scan
  - put
  - etc.
- *./bin/hbase shell SCRIPT.rb*
- Admin
  - DDL
    - create
    - alter, etc.
  - Surgery on sick clusters
    - Operate on regions
    - Force Flushes, etc.


```
durruti:0.18 stack$ ./bin/hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Version: 0.18.1, r707159, Wed Oct 22 12:43:06 PDT 2008
hbase(main):001:0> help
HBASE SHELL COMMANDS:
 alter     Alter column family schema;  pass table name and a dictionary
           specifying new column family schema. Dictionaries are described
           below in the GENERAL NOTES section.  Dictionary must include name
           of column family to alter.  For example, to change the 'f1' column
           family in table 't1' from defaults to instead keep a maximum of 5
           cell VERSIONS, do:

           hbase> alter 't1', {NAME => 'f1', VERSIONS => 5}

 count     Count the number of rows in a table. This operation may take a LONG
           time (Run '$HADOOP_HOME/bin/hadoop jar hbase.jar rowcount' to run a
           counting mapreduce job). Current count is shown every 1000 rows by
           default. Count interval may be optionally specified. Examples:

           hbase> count 't1'
           hbase> count 't1', 100000

 create    Create table; pass table name, a dictionary of specifications per
           column family, and optionally a dictionary of table configuration.
           Dictionaries are described below in the GENERAL NOTES section.
           Examples:

           hbase> create 't1', {NAME => 'f1', VERSIONS => 5}
           hbase> create 't1', {NAME => 'f1'}, {NAME => 'f2'}, {NAME => 'f3'}
           hbase> # The above in shorthand would be the following:
           hbase> create 't1', 'f1', 'f2', 'f3'
           hbase> create 't1', {NAME => 'f1', VERSIONS => 1, TTL => 2592000, \
             BLOCKCACHE => true}
```

Master: aa0-000-8.u.powerset.com:60000

Local logs, Thread Dump, Log Level

Master Attributes

Attribute Name	Value	Description
HBase Version	0.19.1, r49460	HBase version and svn revision
HBase Compiled	Fri Mar 20 11:58:23 PDT 2009, michael	When HBase version was compiled and by whom
Hadoop Version	0.19.2-dev, r0.19	Hadoop version and svn revision
Hadoop Compiled	Fri Mar 20 13:55:44 PDT 2009, michael	When Hadoop version was compiled and by whom
HBase Root Directory	hdfs://coral-dfs.cluster.powerset.com:10000/hbase/aa0-000-8.u.powerset.com	Location of HBase home directory
Load average	43.0	Average load across all region servers. Naive computation.
Regions On FS	32163	The Number of regions on FileSystem. Rough count.

Catalog Tables

Table	Description
-ROOT-	The -ROOT- table holds references to all .META. regions.
.META.	The .META. table holds references to all User Table regions

User Tables

46 table(s) in set.

Table	Description
TestTable	{NAME => 'TestTable', IS_ROOT => 'false', IS_META => 'false', FAMILIES => [{NAME => 'info', BLOOMFILTER => 'false', VERSIONS => '3', COMPRESSION => 'NONE', LENGTH => '2 => 'false'}], INDEXES => []}
biem_10Ktest_c	{NAME => 'biem_10Ktest_c', IS_ROOT => 'false', IS_META => 'false', FAMILIES => [{NAME => 'llsig', BLOOMFILTER => 'false', VERSIONS => '1', COMPRESSION => 'NONE', LENGT BLOCKCACHE => 'false'}, {NAME => 'freq', BLOOMFILTER => 'false', VERSIONS => '1', COMPRESSION => 'NONE', LENGTH => '2147483647', TTL => '-1', IN_MEMORY => 'false', B
biem_10Ktest_meta	{NAME => 'biem_10Ktest_meta', IS_ROOT => 'false', IS_META => 'false', FAMILIES => [{NAME => 'freq', BLOOMFILTER => 'false', COMPRESSION => 'NONE', VERSIONS => '1', LEN BLOCKCACHE => 'false'}], INDEXES => []}
biem_10Ktest_w	{NAME => 'biem_10Ktest_w', IS_ROOT => 'false', IS_META => 'false', FAMILIES => [{NAME => 'llsig', BLOOMFILTER => 'false', VERSIONS => '1', COMPRESSION => 'NONE', LENGT BLOCKCACHE => 'false'}, {NAME => 'freq', BLOOMFILTER => 'false', VERSIONS => '1', COMPRESSION => 'NONE', LENGTH => '2147483647', TTL => '-1', IN_MEMORY => 'false', B
copyofenwikilive	{NAME => 'copyofenwikilive', IS_ROOT => 'false', IS_META => 'false', FAMILIES => [{NAME => 'alternate_title', BLOOMFILTER => 'false', COMPRESSION => 'NONE', VERSIONS => ' BLOCKCACHE => 'false'}, {NAME => 'anchor', BLOOMFILTER => 'false', COMPRESSION => 'NONE', VERSIONS => '3', LENGTH => '2147483647', TTL => '-1', IN_MEMORY => 'false' BLOOMFILTER => 'false', COMPRESSION => 'NONE', VERSIONS => '3', LENGTH => '2147483647', TTL => '-1', IN_MEMORY => 'false', BLOCKCACHE => 'false'}, {NAME => 'page', B

Getting Started

- No one reads the *Getting Started* section
  - First thing on our home page, first thing in javadoc
- File descriptors
  - ulimit -n >>>> 1024
- Xceiver count in Hadoop
  - Weird HBase failures and datanode log has "*xceiverCount 258 exceeds the limit of concurrent xcievers 256*"
  - Make *dfs.datanode.max.xcievers* [sic] > 256 ... 4-10x
- DFSClient timeout or *"No live nodes contain current block"*
  - HBase sees DFSC errors others don't because long-running
  - Set *dfs.datanode.socket.write.timeout* to 0 at extreme
- Undo *DBA-Think* doing HBase schema design
  - Locality, physical layout, key-design
- “Commodity Hardware” != old linux box found up in attic
  - 4-8 cores, 4+ Gigs of RAM
  - EC2, less headache if X-large instances

History

- *10-11/2006* Powerset interest in Bigtable
  - Chad Walters & Jim Kellerman
  - Paper published around this time
- 02/2007 Mike Cafarella drive-by code dump
  - Jim cleans it up. Added as Hadoop contrib
- 10/2007 0.15.0 Hadoop
  - First "*usable*" HBase
- *12/18/2007* First HBase User Group @ rapleaf
- 01/16/2008 Hadoop TLP, HBase subproject
  - HBase own mailing lists, area in JIRA, website
- 03/19/2009 HBase 0.19.1
  - 10th HBase release

Project Status: Current Release

0.19.0 released 01/2009

- 185 fixes since 0.18.0
- HBase Metrics (Ganglia)
  - Requests/Regions/StoreFiles/Memory
- Performance improvements:
  - Roughly 3X writing, 7X scanning
  - Random-reads 1-4X (dependent on cache-size/hits)
  - General HDFS speedup
  - HBaseRPC -- codes for param and method names
  - Pre-fetch when Scanning
  - Batching writes
  - Server-caching of small blocks (16k) of HDFS files
- Open-cluster surgery tools
  - Manual close, split, flush regions

0.19.1 released 03/19/2009

- 40 fixes

Project Status: Current Development

0.20.0: Performance and Zookeeper Integration

On Zookeeper Integration (HA HBase/No SPOF)

- Moving Master functionality to ZK
  - General cluster state
  - Server leases
  - Root region location
- Multiple Masters, election on failure
  - TODO: Regionservers moving to new master, smoothly
- HBase manages ZK Cluster
  - ...unless pointed at existing ZK Cluster
- Future: masterless HBase cluster?
  - Not for 0.20.0 HBase

Project Status: Current Dev. Contd.

Performance Goal: fast enough to serve a website

- Website Use Cases
  1. Random Seek, then Scan 10-1000 rows
  2. Get all of a row at a random offset
- Dev to date:
  - New block-based File Format, hfile
  - Looked at using TFile, HADOOP-3315
    - Too complicated, too many streams
  - hfile more performant than TFile and Sequencefile
    - Opening MapFiles on each Scanner open
      - Couldn't seek+scan < ~200ms, now ~20ms
- Dev TODO:
  - Byte-arrays everywhere/Zero-copy RPC to FileSystem
  - RawComparators rather than Comparators
  - Cut object creation
  - Finish up Cell and block caches, "in-memory" tables
    - Built-in "memcached" -- remove a layer

Project Status: Current Dev. Contd.

Full-disclosure slide...

- Still no sync/flush/append in HDFS
- So DATA LOSS on regionserver crash, still
- See hadoop-4379, 5332, etc.
  - Hopefully Hadoop 0.21.0

Project Status: Community

- 5 Committers
  - JimK, J-D Cryans, Andrew Purtell, and Stack (Bryan Duxbury dormant)
- Recent Significant Contributors
  - Ryan Rawson new hfile (was rfile)
  - John Gray and Erik Holstad, Caching, Backup
  - Sishen Freecity/Brian Beggs look after REST
  - Clint Morgan transactional hbase (0CC), etc

- 6User Group Meetups to date
  - SOCAL HBackathon, January 30th
    - 10-12 attendees over two days of discussions & code
  - Berlin Hadoop Meetup -- March 5th, 2009
    - Lars George of WorldLingo
- HBase Extensions
  - ORMs: *pigi, Parhely, OHM*
  - *HBase-Writer -- IA Heritrix crawling to HBase table*
  - *datastore "Implementation of the Google App Engine Datastore in Java"*

Powered-by: 1 of 2

- *Mahalo.com* : "the worlds first human-powered search engine"
  - All markup that powers the wiki is stored in HBase.
  - Mahalo's in-house editors produce a lot of revisions per day, not working well in a RDBMS (MediaWiki in MySQL).
  - Right now it's at something like 6 million items in HBase.
- *WorldLingo* host their Multilingual Archive in HBase.
  - Scan with MapReduce to machine-translate and index documents
  - Concurrent updates
    - MS Office provider since 2001
  - User-facing, memcached in front of hbase
  - Currently > tens of millions, aiming for 450M.
  - User since November 2007, HBase 0.15.1

Powered-by: 2 of 2

- *Streamy*
  - All data for website in HBase
    - Replaced an RDBMS
      - Got so big 10M rows, denormalized and put all in one table so no joins
      - Writes still too slow so partitioned
      - Then tables by-day
      - Eventually, "...lost in admin..."
  - Caching in front of HBase, "..as you'd do with any RDBMS..."
- *Others*
  - *OpenPlaces*
  - Groups at *Adobe* and *Yahoo!*
  - *Powerset(Microsoft):*
    - 110 node cluster with ~50 tables of many millions of rows each

Thanks

- stack@apache.org
- Visit ***hbase.org*** : mailing lists, source
- Join the *#hbase* channel up on *irc.freenode.net*