HBase: structured storage of sparse data for Hadoop

Jim Kellerman

Powerset, Inc.

jim@powerset.com jimk@apache.org

Topics

- Introduction, brief history
- Concepts
- Architecture
- Client API
- Tools
- Project Status
- Questions And Answers

Brief History

- 6/06 Mike Cafarella posts on Hadoop mailing lists
- 10-11/06 Powerset interest in Bigtable, recruit others, some design documents
- 01-02/07
  - Powerset resumes work
  - Mike Cafarella provides initial code base
- 04/07 Michael Stack joins Powerset
- 10/07 First usable version of HBase in Hadoop 0.15.0 release

Concepts

- Data addressed by row/column/version key
  - Version can be a specified by client
    - default is `System.currentTimeMillis`
  - Data is stored column-oriented rather than row-oriented
    - More space efficient – nulls are free
    - Better compression – data is similar
- Updates lock entire row
  - Don't need to update every column
  - Locks never span rows

Concepts (contd.)

Columns (aka column families):

- Values are byte[]
- Most options on per-column basis:
  - # of versions, compression, bloom filters, maximum value length
- Fixed at table creation
  - Can be added/dropped if table is off-line
  - Family members can be created/deleted at any time

Concepts (contd.)

Example: Web Crawl data

		Column Names (families)			
Row Key	**Version #**	**contents:**	**text:**	**meta:**	
<url1>	crawl-time (sorted descending)	original document	extracted text	meta:mime	text/html
				meta:refresh	daily
				meta:spam	(score value)
				meta:language	en-uk
				meta:encoding	ISO-Latin1
	...	...	...	meta:mime	text/html
				meta:encoding	ISO-Latin1
<url2>	...	...	...	...	

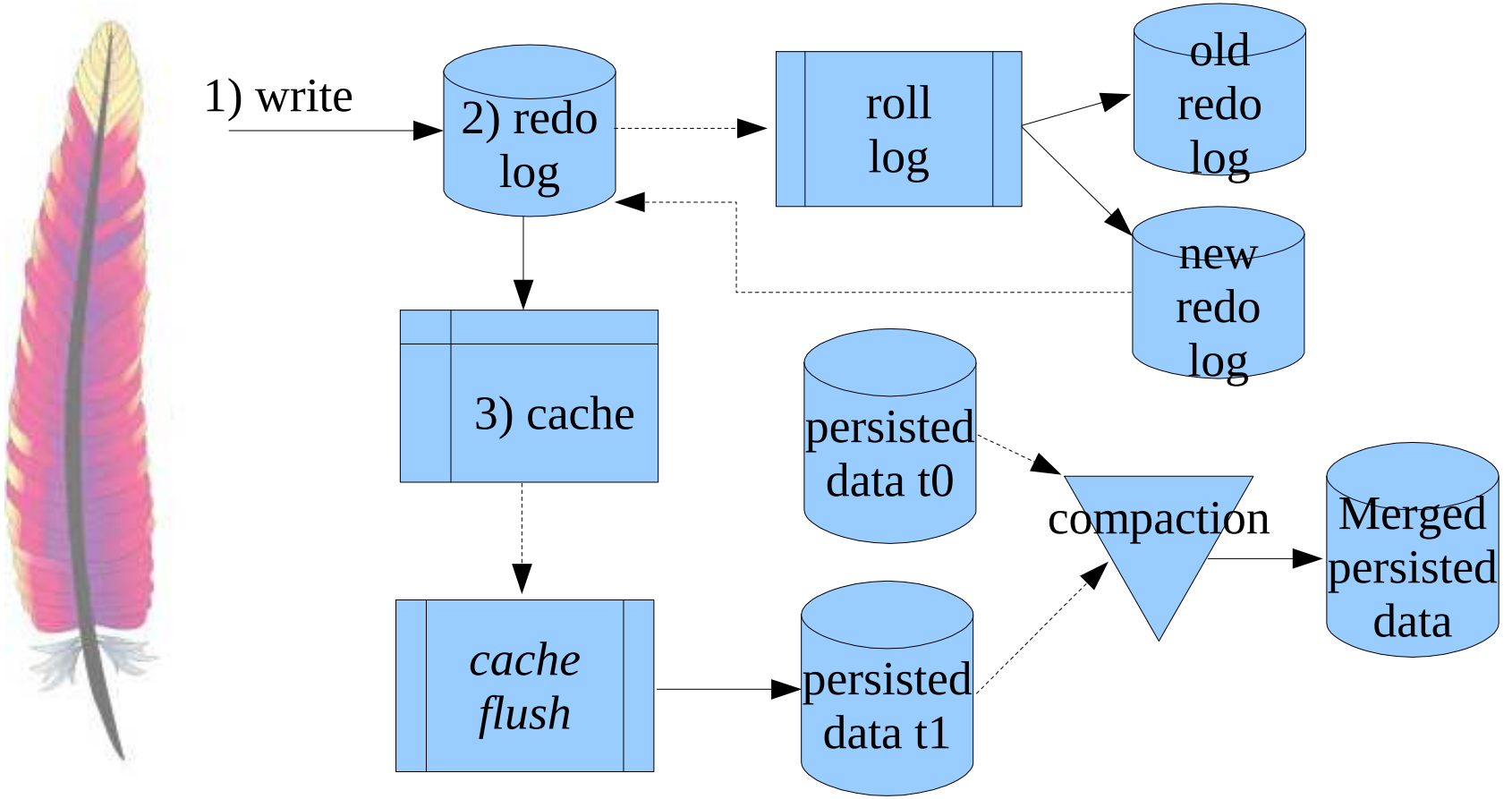
HBase Concepts (contd.)

- Tables are stored in regions
  - A region is a row range for the table [start-key:end-key)
  - When regions get "too big" they are split
    - The two new regions get ½ the row range of the parent region
      - » One gets lower half of row range
      - » One gets upper half of row range
    - Splits are instantaneous
  - Each column family is stored in an HStore
    - Each region has one HStore per column family

HBase Concepts (contd.)

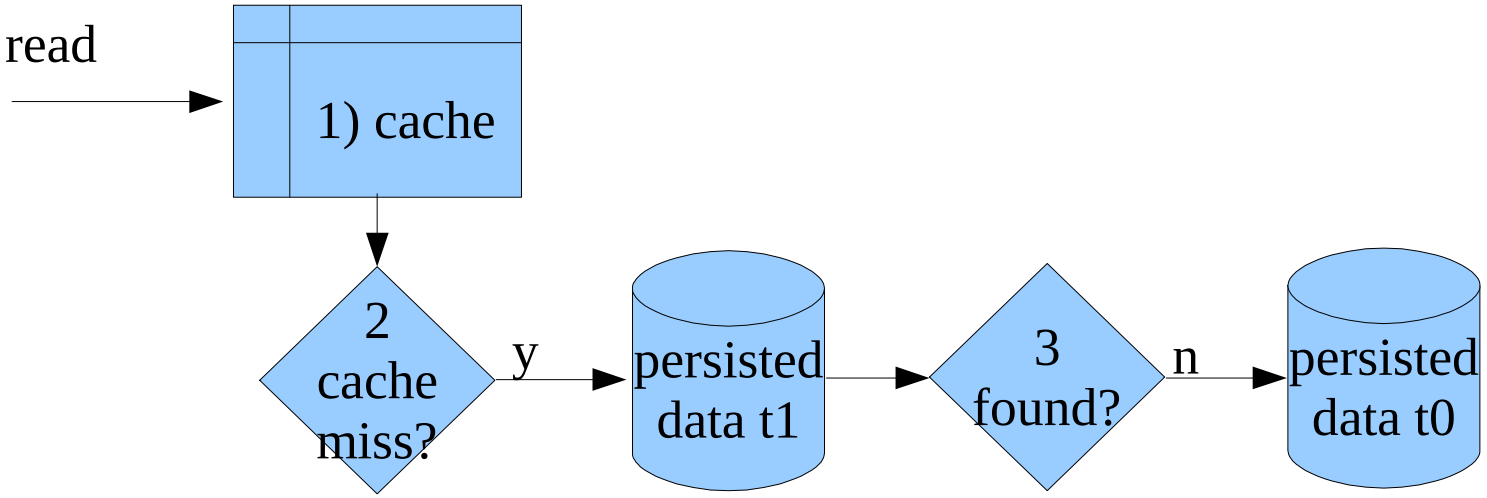
- When a write occurs:
  - It is written to a write ahead log
  - It is cached in memory
  - Periodically, the cache is written to disk, creating a new file in each HStore for the columns being flushed
    - A compaction occurs when the number of files in an HStore exceeds a threshold
      - » Compactions are done in the background
  - Periodically, the log file is closed and a new one created
    - Old log files are garbage collected

Concepts (contd.)



Concepts (contd.)

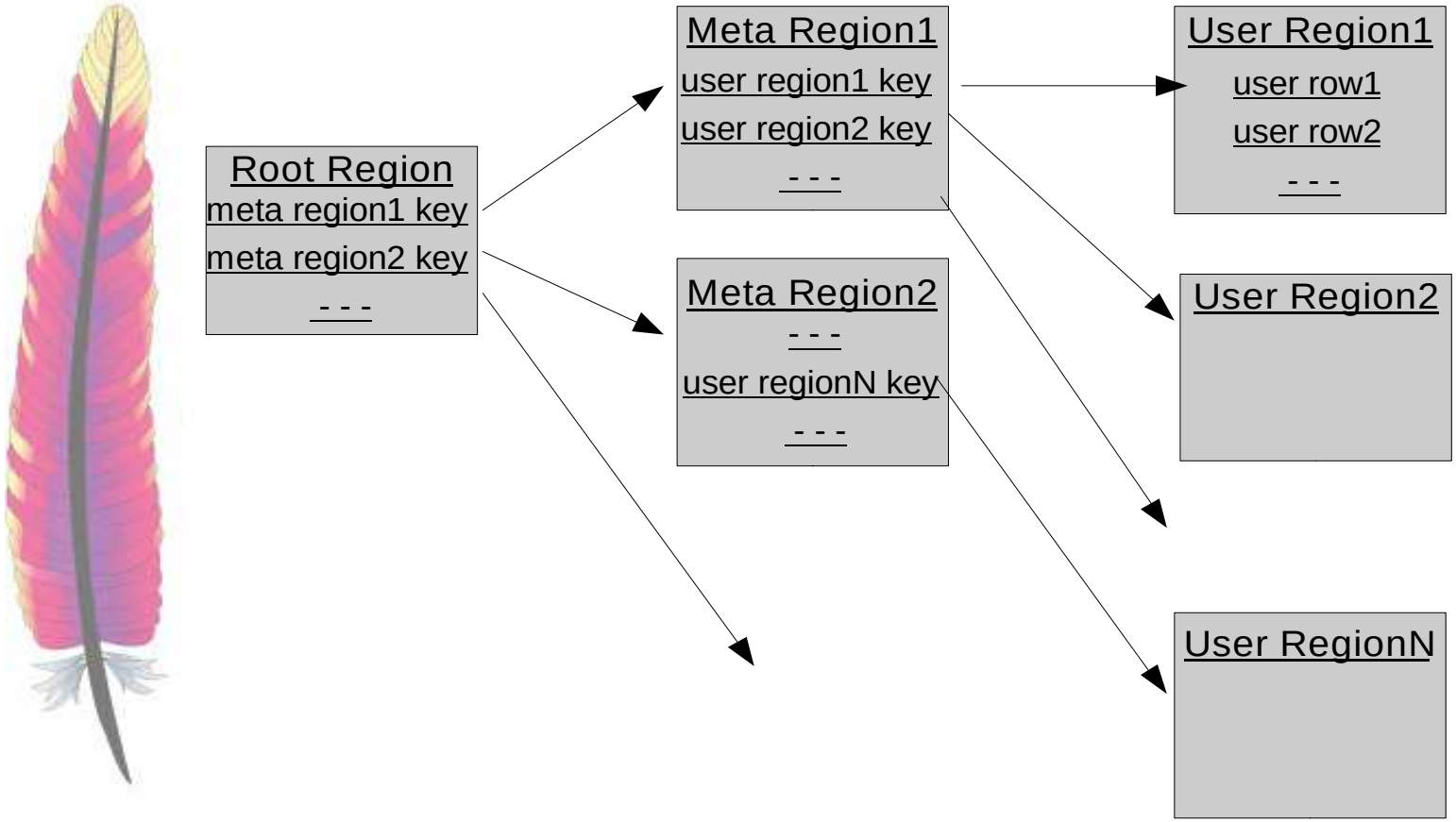
- When a read occurs:
  1)Check for data in cache
  2)Look for data in persisted data, from newest to oldest.



HBase Concepts (contd.)

- The location of all user regions is stored in the META region
  - Each META row maps one user region
  - Each META region can map about 64K regions
- All the META regions are mapped by one ROOT region
  - ROOT and META can map about 8×10^9 user regions
  - With current region size of 64MB, about 10^{18} bytes of data can be stored

Concepts (contd.)



HBase: Architecture

- There are three major components:
  - Master server
  - Region server
  - Client API

HBase Architecture: Master Server

- Assigns regions to region servers
  - The ROOT region is assigned first
- Scans ROOT region to find META regions, assigns them to region servers
- Scans META regions to find user regions, assigns them to region servers
- Reassigns regions for load balancing or if region server fails
- Tells client where ROOT region is located

Architecture: Region Server

- Server threads handle client requests
- Main thread is master heart beat loop
- Other threads:
  - Process long running operations resulting from master heart beat response
  - Check to see if regions need to be split
  - Check to see if cache needs to be flushed
  - Check to see if log needs to be rolled
  - One thread per client request (leases)

Architecture: HRegion

- One HRegion per table fragment (row range)
  - In memory cache of recent writes
  - One HStore object per column
    - Finds and reads data from HDFS
    - May manage many files (one is created per cache flush)
    - Performs compaction if too many files
    - Performs split operation for a single column

Client API

- Create new HTable object to open a table
  - Client locates ROOT region from master
  - Client reads (and caches) ROOT region to locate META servers
  - Client reads (and caches) META information for the table being opened
- Client requests are sent directly to region servers. Master is not involved
- Administrative functions to manipulate tables and columns

Client API (contd.)

- Read
  - Specific row/column pair, specific row – all columns
    - Most recent version
    - Specific version
    - N versions
    - All versions
  - Scan multiple rows
- Write
  - All row mutations are atomic
  - Can write multiple columns in single update

Client API Examples

```
// Open table
HTable table = new HTable(conf, tableName);

// Storing data
long writeid = table.startUpdate(row);
table.put(writeid, columnName1, bytes);
table.put(writeid, columnName2, bytes);
table.delete(writeid, columnName3);
table.commit(writeid);

// Reading data
byte[] data = table.get(row, columnName1);
```

Client API Examples (contd.)

```
// Open scanner
HScannerInterface scanner = table.obtainScanner(cols, new Text());

try {
  SortedMap<Text, byte[]> values = new TreeMap<Text, byte[]>();
  HStoreKey currentKey = new HStoreKey();
  while (scanner.next(currentKey, values)) {
    // row: currentKey.getRow(), version: currentKey.getTimestamp()
    for (Map.Entry<Text, byte[]> e: values.entrySet()) {
      // columnName: e.getKey(), value: e.getValue()
    }
  }
} finally {
  scanner.close();
}
```

HBase: Tools

- HBase shell
- Web Interface

Tools: HBase Shell

```
durruti$ ./bin/hbase shell
Hbase Shell, 0.0.2 version.
Copyright (c) 2007 by udanax, licensed to Apache Software Foundation.
Type 'help;' for usage.
Hbase> create table 'test' ('test');
Hbase> insert into 'test' ('test:test') values ('some old value')
   where row="test_row";
Hbase> select * from 'test';
+----------+-------------+-------------------------+
| Row      | Column      | Cell                    |
+----------+-------------+-------------------------+
| test_row | test:test   | some old value          |
+----------+-------------+-------------------------+
Hbase>
```

Tools: Web Interface

Master: 127.0.0.1:60000

HBASE HQL, Local logs, Thread Dump

Master Attributes

Attribute Name	Value
Filesystem	file:///
Hbase Root Directory	/tmp/hadoop-stack/hbase

Online META Regions

Name	Server
-ROOT-	208.76.47.66:56086
.META.,,1	208.76.47.66:56086

Tables

Name	Descriptor
test	name: test, families: {test:={name: test, max versions: 3, compression: NONE, in memory: false, max length: 2147483647, bloom filter: none}}

1 table(s) in set

Region Servers

Address	Start Code	Load
208.76.47.66:56086	-4128465680919009177	requests: 0 regions: 3

Tools: Web Interface (contd.)

Region Server: 208.76.47.66:56086

Local logs, Thread Dump

Region Server Attributes

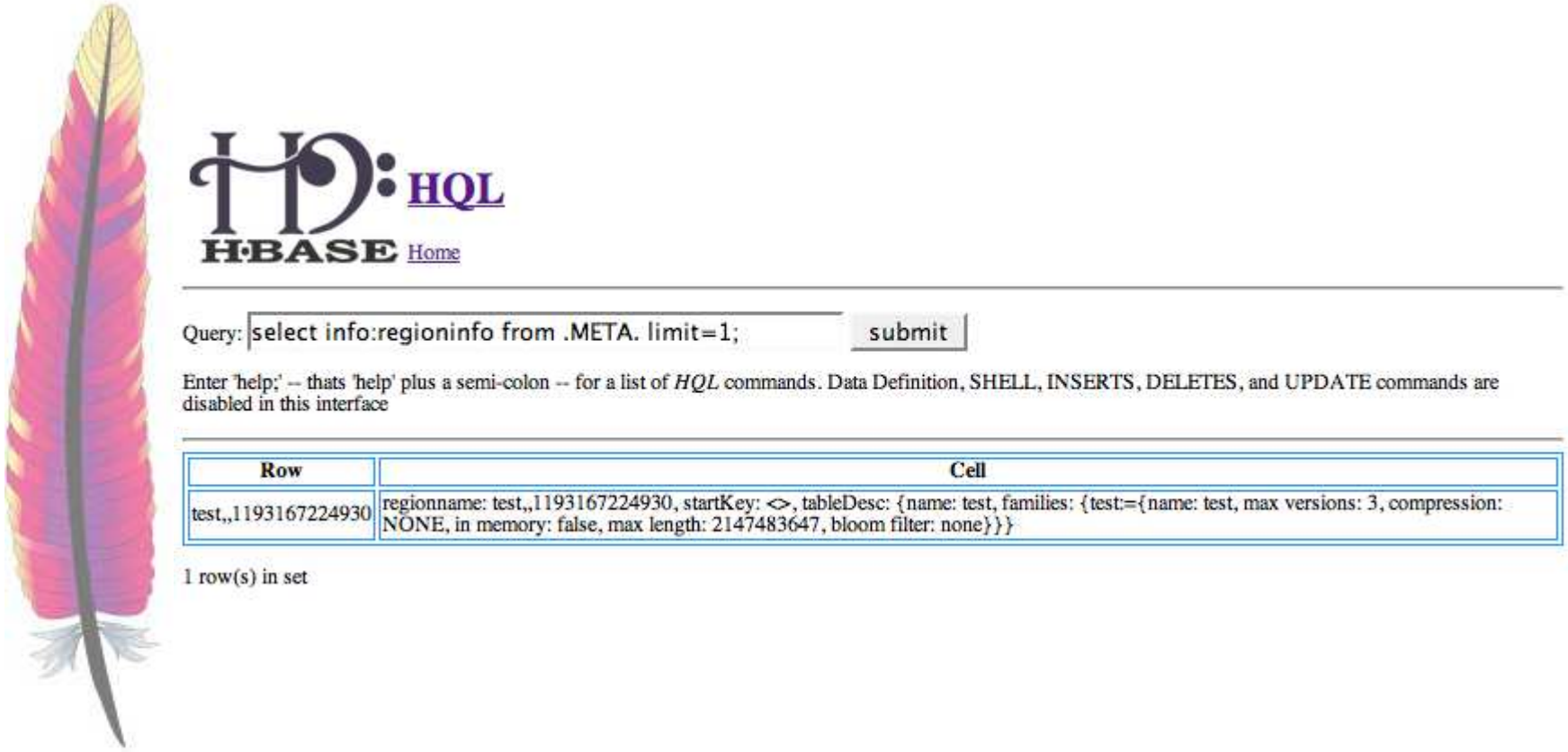
Attribute Name	Value
Load	requests: 0 regions: 3

Online Regions

Region Name	Start Key	End Key
-ROOT-,,0		
.META.,,1		
test,,1193167224930		

Region names are made of the containing table's name, a comma, the start key, a comma, and a randomly generated region id. To illustrate, the region named *domains,apache.org,5464829424211263407* is party to the table *domains*, has an id of *5464829424211263407* and the first key in the region is *apache.org*. The *-ROOT-* and *.META.* 'tables' are internal sytem tables. The -ROOT- keeps a list of all regions in the .META. table. The .META. table keeps a list of all regions in the system. The empty key is used to denote table start and table end. A region with an empty start key is the first region in a table. If region has both an empty start and an empty end key, its the only region in the table. See Hbase Home for further explication.

Tools: Web Interface (contd.)



HBASE HQL

HBASE Home

Query: select info:regioninfo from .META. limit=1; submit

Enter 'help;' -- thats 'help' plus a semi-colon -- for a list of *HQL* commands. Data Definition, SHELL, INSERTS, DELETES, and UPDATE commands are disabled in this interface

Row	Cell
test,,1193167224930	regionname: test,,1193167224930, startKey: <>, tableDesc: {name: test, families: {test:={name: test, max versions: 3, compression: NONE, in memory: false, max length: 2147483647, bloom filter: none}}}

1 row(s) in set

Project Status

- First "usable" release of HBase included in Hadoop-0.15.0
  - However data loss possible without Hadoop "append" support.
- To do:
  - Build community: users, contributors
  - Documentation and ease-of-use features
  - Performance analysis
  - ZooKeeper integration
  - More Monitoring

Project Status (contd.)

- Several key contributions to date.
  - Map/Reduce connector
  - HBase shell
    - Relational operators (in progress)
  - Restructure so applications can access multiple tables simultaneously.

Interested?

- Get involved!
  - Contributions welcome!
  - Follow email lists and Jira

Questions? And Answers!

References:

"Bigtable: A Distributed Storage System for Structured Data"

http://labs.google.com/papers/bigtable.html

The HBase Wiki at Apache.org

http://wiki.apache.org/lucene-hadoop/Hbase

The #hbase IRC chat room at freenode.net

The Hadoop mailing lists