Intro to HBase

Alex Baranau, Sematext International, 2012

About Me

- Software Engineer at Sematext International
- http://blog.sematext.com/author/abaranau
- @abaranau
- http://github.com/sematext (abaranau)

Agenda

- What is HBase?
- How to use HBase?
- When to use HBase?

What is HBase?

What: HBase is...

Open-source non-relational distributed column-oriented database modeled after Google's BigTable.

- Think of it as a sparse, consistent, distributed, multidimensional, sorted map:
  - labeled tables of rows
  - row consist of key-value cells:

(row key, column family, column, timestamp) -> value

What HBase is NOT

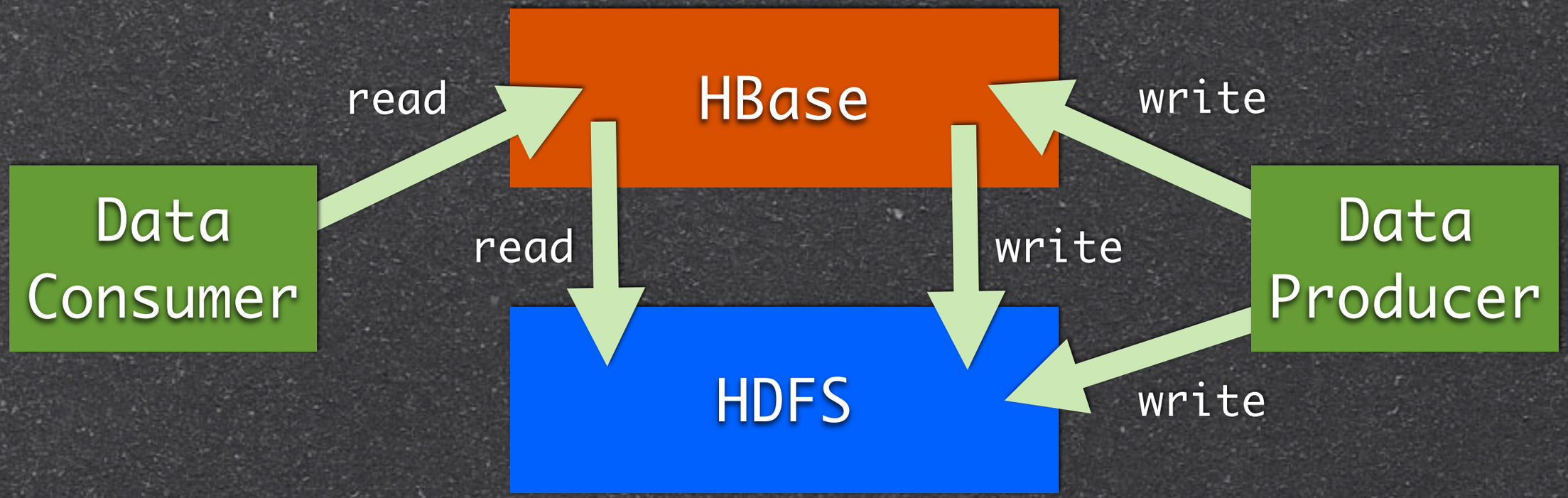
- Not an SQL database
- Not relational
- No joins
- No fancy query language and no sophisticated query engine
- No transactions out-of-the box
- No secondary indices out-of-the box
- Not a drop-in replacement for your RDBMS

What: Features-1

- Linear scalability, capable of storing hundreds of terabytes of data
- Automatic and configurable sharding of tables
- Automatic failover support
- Strictly consistent reads and writes

What: Part of Hadoop ecosystem

- Provides realtime random read/write access to data stored in HDFS



What: Features-2

- Integrates nicely with Hadoop MapReduce (both as source and destination)
- Easy Java API for client access
- Thrift gateway and REST APIs
- Bulk import of large amount of data
- Replication across clusters & backup options
- Block cache and Bloom filters for real-time queries
- and many more...

How to use HBase?

How: the Data

- Row keys uninterpreted byte arrays
- Columns grouped in columnfamilies (CFs)
- CFs defined statically upon table creation
- Cell is uninterpreted byte array and a timestamp

Rows are ordered and accessed by row key

Different data separated into CFs

All values stores as byte arrays

Row Key	Data
Minsk	geo:{'country':'Belarus','region':'Minsk'} demography:{'population':'1,937,000'@ts=2011}
New_York_City	geo:{'country':'USA','state':'NY'} demography:{'population':'8,175,133'@ts=2010, 'population':'8,244,910'@ts=2011}
Suva	geo:{'country':'Fiji'}

Rows can have different columns

Cell can have multiple versions

Data can be very "sparse"

How: Writing the Data

- Row updates are atomic
- Updates across multiple rows are NOT atomic, no transaction support out of the box
- HBase stores N versions of a cell (default 3)
- Tables are usually “sparse”, not all columns populated in a row

How: Reading the Data

- Reader will always read the last written (and committed) values
- Reading single row: Get
- Reading multiple rows: Scan (very fast)
  - Scan usually defines start key and stop key
  - Rows are ordered, easy to do partial key scan

Row Key	Data
‘login_2012-03-01.00:09:17’	d:{‘user’:‘alex’}
...	...
‘login_2012-03-01.23:59:35’	d:{‘user’:‘otis’}
‘login_2012-03-02.00:00:21’	d:{‘user’:‘david’}

- Query predicate pushed down via server-side Filters

How: MapReduce Integration

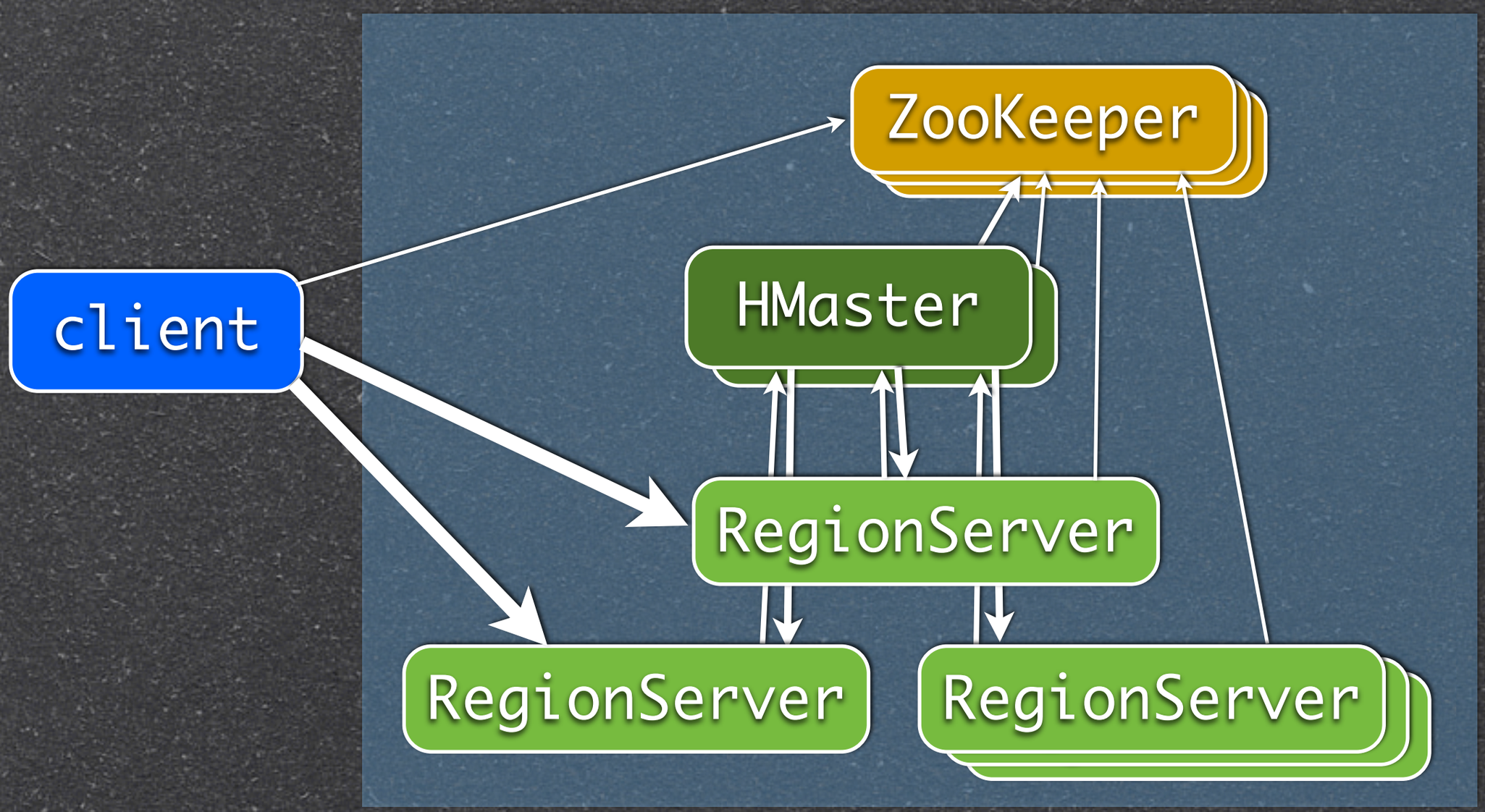
- Out of the box integration with Hadoop MapReduce
  - Data from HBase table can be source for MR job
  - MR job can write data into HBase
  - MR job can write data into HDFS directly and then output files can be very quickly loaded into HBase via “Bulk Loading” functionality

How: Sharding the Data

- Automatic and configurable sharding of tables:
  - Tables partitioned into Regions
    - Region defined by start & end row keys
    - Regions are the “atoms” of distribution
- Regions are assigned to RegionServers (HBase cluster slaves)

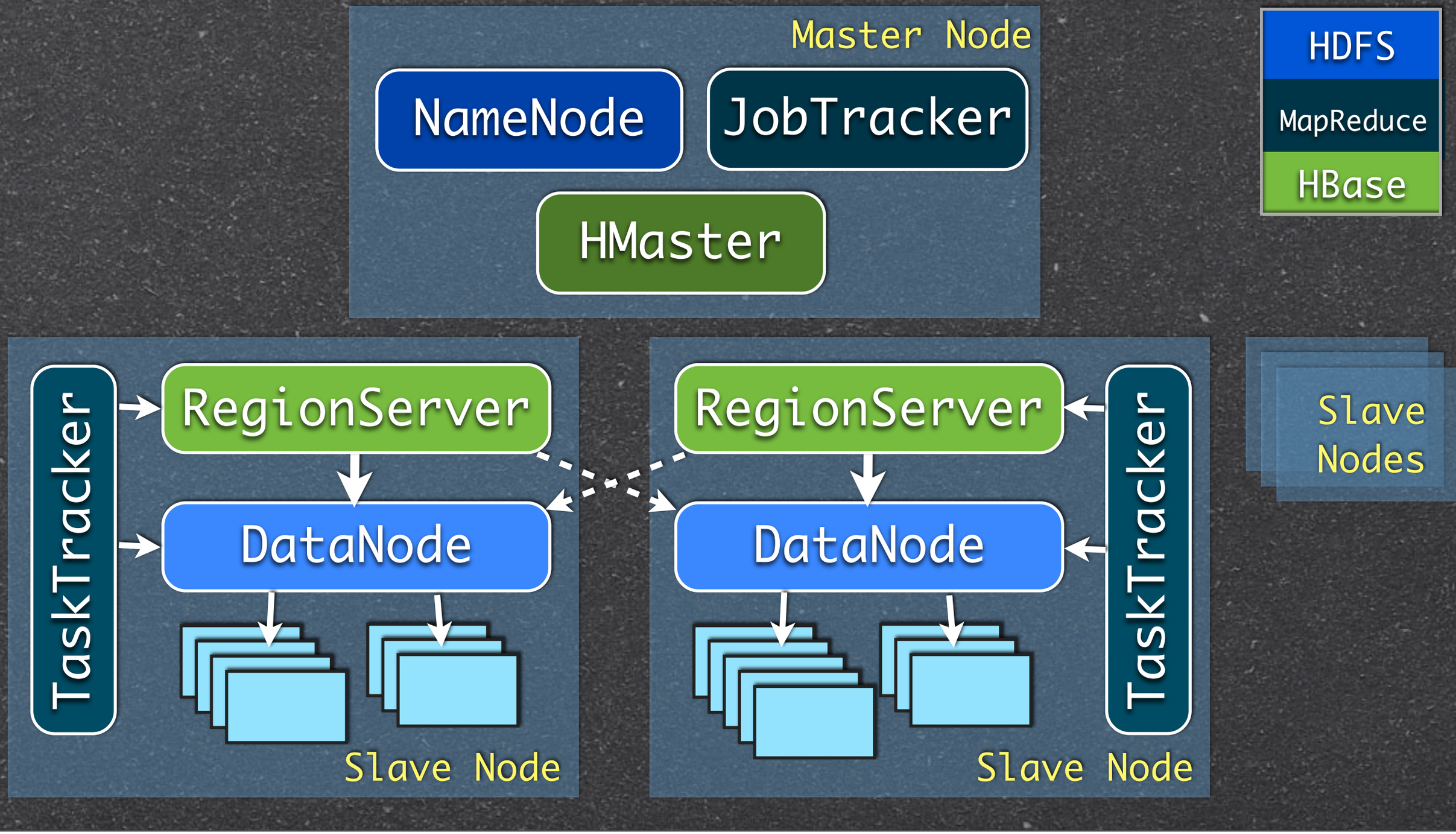
How: Setup: Components

- HBase components



How: Setup: Hadoop Cluster

- Typical Hadoop+HBase setup



How: Setup: Automatic Failover

- DataNode failures handled by HDFS (replication)
- RSs failures (incl. caused by whole server failure) handled automatically
  - Master re-assignes Regions to available RSs
- HMaster failover: automatic with multiple HMasters

When to Use HBase?

When: What HBase is good at

- Serving large amount of data: built to scale from the get-go
- fast random access to the data
- Write-heavy applications*
- Append-style writing (inserting/overwriting new data) rather than heavy read-modify-write operations**

* clients should handle the loss of HTable client-side buffer
** see https://github.com/sematext/HBaseHUT

When: HBase vs ...

- Favors consistency over availability
- Part of a Hadoop ecosystem
- Great community; adopted by tech giants like Facebook, Twitter, Yahoo!, Adobe, etc.

When: Use-cases

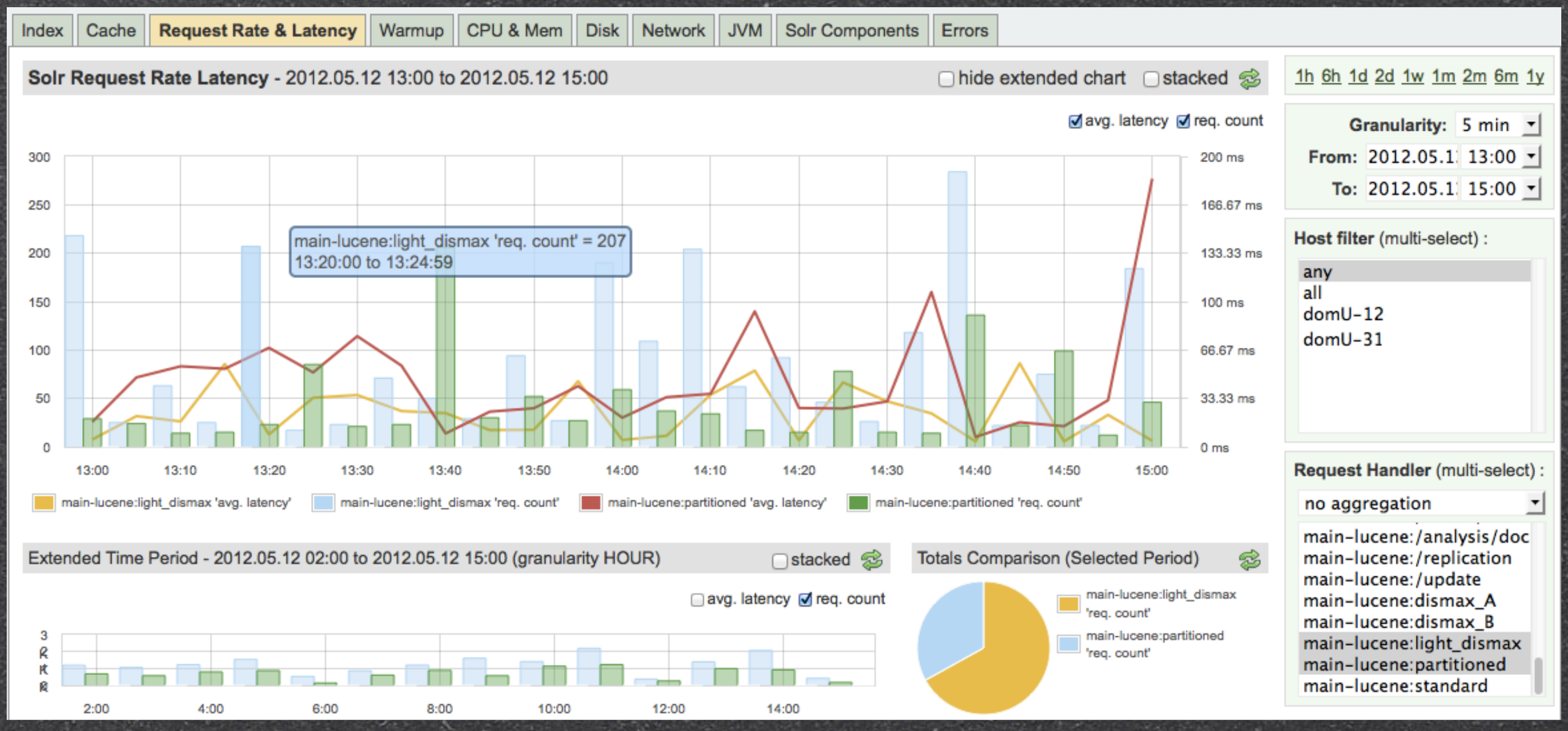
- Audit logging systems
  - track user actions
  - answer questions/queries like:
    - what are the last 10 actions made by user?
      row key: userId_timestamp
    - which users logged into system yesterday?
      row key: action_timestamp_userId

When: Use-cases

- Real-time analytics, OLAP
  - real-time counters
  - interactive reports showing trends, breakdowns, etc
  - time-series databases

When: Use-cases

- Monitoring system example



When: Use-cases

- Messages-centered systems
  - twitter-like messages/statuses
- Content management systems
  - serving content out of HBase
- Canonical use-case: webtable (pages stored during crawling the web)
- And others

Future

- Making stable enough to substitute RDBMS in mission critical cases
- Easier system management
- Performance improvements

Qs?

(next: Intro into HBase Internals)

Sematext is hiring!