Rails on HBase

Zachary Pinter and Tony Hillerson
RailsConf 2011
What we will cover
H·BASE

What is it?
What are the tradeoffs that HBase makes?
Why HBase is probably the wrong choice for your app
Why HBase might be the right choice for your app
How to use HBase with Rails
What we won’t cover
This is not a deep dive
This is not a guide to setting up a production HBase cluster

*See http://cloudera.com
So...

What is HBase?
HBase is based on Bigtable
HBase Grew out of Hadoop
Core Concepts
Distributed Column Oriented Database
### Tables

<table>
<thead>
<tr>
<th>key</th>
<th>core:link</th>
</tr>
</thead>
<tbody>
<tr>
<td>llb0ga</td>
<td><a href="http://google.com">http://google.com</a></td>
</tr>
</tbody>
</table>
# Column Families

<table>
<thead>
<tr>
<th>key</th>
<th>core:link</th>
<th>stats:hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>llb0ga</td>
<td><a href="http://google.com">http://google.com</a></td>
<td>100504</td>
</tr>
</tbody>
</table>
Column Families

Must be defined at creation or added to schema later

Should group data accessed and sized similarly
Columns

Can be dynamically added given that the family exists

Stored as a byte array - do what you need with it
Column Families: Data Locality
Regions

master

Regionserver

Regionserver

Regionserver
Main Operations

Get
Put
Incr
Delete
Scan
JRuby based shell
ruby-1.9.2-p180 :002 > status
1 servers, 0 dead, 1.0000 average load

ruby-1.9.2-p180 :003 > (1..10).each{ status }
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load
1 servers, 0 dead, 3.0000 average load

=> 1..10

ruby-1.9.2-p180 :004 >
API

REST
Thrift
Apache Avro
Java API
What are the tradeoffs that HBase makes?

NoSQL = Tradeoffs
# Highly Distributed

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Built-in sharding</td>
<td>• Makes joins difficult/impossible</td>
</tr>
<tr>
<td>• No single server holds all the data</td>
<td>• Limited real-time queries/sorting</td>
</tr>
<tr>
<td>• Different servers operate on different slices of the data</td>
<td></td>
</tr>
</tbody>
</table>
Sorted Row Keys

Bytearray keys
Any kind of data you want
Sorted in byte order
All row access through the key
# Sorted Row Keys

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Real-time random lookups by key</td>
<td>• Conflates unique identifier with default sort</td>
</tr>
<tr>
<td>• Real-time range queries</td>
<td>• Keys are an integral part of schema design, requiring extra consideration</td>
</tr>
</tbody>
</table>
## Map / Reduce

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Scales linearly: more servers = faster response times</td>
<td>• Scans every row per query *</td>
</tr>
<tr>
<td>• Operates on huge datasets (billions of rows)</td>
<td>• Not real-time (batch operation)</td>
</tr>
<tr>
<td>• Works well with dynamic/non-uniform schemas</td>
<td></td>
</tr>
</tbody>
</table>
HBase vs. X

http://nosql-database.org

http://www.sevenweeks.org/
Seven Databases in Seven Weeks
Eric Redmond

Don’t Miss His Talk!!
Why HBase is probably the wrong choice for your app
HBase Limitations

LIMITATIONS

Until you spread your wings, you'll have no idea how far you can walk.

www.despair.com
HBase Limitations

- Limited real-time queries (key lookup, range lookup)
- Limited sorting (one default sort per table)
- Limited transactions
- Manual indexing
- Joins/Normalization
- Storage of large binary data *
HBase Requirements

- At least 5 Servers (Zookeeper, Region Server, Name Nodes, Data Nodes, etc)
- Memory Intensive (2-4gb bar minimum per server, depending on server role)
- CPU Intensive
- Rough EC2 Cost:

  5 c1.xlarge reserved for 1 year
  = over $850/month
  (not counting bandwidth!)
Why HBase might be the right choice for your app
Large data sets

Ability to aggregate and analyze billions of rows

Starts to shine when relational databases break down

Is your db already sharded?
Are you using SQL queries that take hours to run?
Hadoop Integration

HDFS for storing files

Native Map/Reduce
Supports data locality
Doesn’t require data to be transferred
Flexible schema

Supports non-heterogeneous data through column families with mixed key/value pairs

Ability to refactor the data store via map/reduce (beats a multi-day ALTER TABLE command)
Rails on HBase
hbase-stargate

https://github.com/greglu/hbase-stargate

Uses Stargate, HBase REST server
Rhino

https://github.com/sqs/rhino/

Thrift API

Seems to be abandoned?
Massive Record

https://github.com/CompanyBook/massive_record

It works!

Uses Thrift API
Kwisatz: An URL Shortener
Demo: JRuby Scripts
Demo:
Rails Front-End
Source

Demo source code

github.com/effectiveui/jruby-hbase-demo
github.com/effectiveui/rails-hbase-massive-record
Getting Set Up

(Mac) brew install hbase

Cloudera VM
Recap
Where Would I Use HBase?
Where is HBase's Sweet Spot
Where Would I Use Rails With HBase?
Questions?
Thank You!

Zachary: @zpinter
Tony: @thillerson
EffectiveUI.com