Real World HBase + Scala + Hadoop

Eugene Marinelli & Quinn Slack
{eugene, quinn} @blendlabsinc.com

So you’re starting your own company...

...and you’re solving a really, really hard problem

...you have a ton of data, need to handle a lot of users, and want to perform heavy computation over the data

What do you use?

HBase

- Based on Google BigTable
  - Google search, Gmail, Maps, etc.
- Used by **Facebook** (messages), **Twitter** (offline analysis), **Greplin, StumpleUpon,** etc.
- Distributed by nature
  - Only makes sense if you have a ton of data

HBase Data Model

- Column families hold data that is accessed together
- Scanning rows sequentially is fast
- Each value's "address" is a (row key, column family, column) combo
  - ex. "alice@stanford.edu" is stored at (alice, info: email)

	Column Family (specified)	**info**		**group**		
	Column (arbitrary)	name	email	Stanford Alumni	Slider Bar	Yogurtland
Rows	alice	Alice Smith	alice@stanford.edu	1	1	
	bob	Bob Smith	bob@stanford.edu	1		1

Source: http://internetmemory.org/en/index.php/synapse/understanding_the_hbase_data_model

...and Scala + Hadoop

Scala

- Concise
- Typed
- Extensible
- Functional

Hadoop

- Scalable Computation
- Flexible
- Distributed

Real World Example

Steps

1. Define the data model
2. Load the data
3. Access it in HBase
4. Analyze it using MapReduce

Modeling

Scala case class

```
Person(
  id: String,
  name: String,
  likes: List[Like]
)

case class Like(
  id: String,
  name: String
)
```

HPaste (simplified) class **PersonTable** {

```
  val info = family("info")
  val name = column(info, "name")
  val like = family("like")
}
```

Pulling data

...directly from JSON into Scala case classes:

```
object Graph {
  private def apiURL(id: String, path: String, limit: Int, fields: String) =
    dispatch.url("https://graph.facebook.com/" + id + "/" + path) <<? Map(
      "fields" -> fields, "access_token" -> accessToken, "limit" -> limit.toString
    )
  def getFriends(id: String = "me", limit: Int = 20): List[Person] = {
    val url = apiURL(id, "friends", fields = "id,name", limit = limit)
    dispatch.Http(url ># { json =>
      (json \ "data").extract[List[Person]]
    })
  }
}
```

Accessing HBase

```
object PersonHBaseCollection {
  val Me = "me"

  def put(person: Person) =
    PersonSchema.PersonTable
      .put(person.id)
      .value(_.name, person.name)
      .valueMap(_.like, person.likes.map(like => like.id -> like.name).toMap)
      .execute()

  def me = get(Me).getOrElse(throw new Exception("me not found."))

  def get(id: String): Option[Person] =
    PersonSchema.PersonTable.query2
      .withKey(id)
      .singleOption()
      .map(_.toPerson)
}
```

MapReduce analyses

```
class TopLikesMapper extends FromTableBinaryMapperFx(PersonTable) {
  val person = row.toPerson
  for (like <- person.likes) {
    val keyOutput = makeWritable(_.writeUTF(like.name))
    val valueOutput = makeWritable(_.writeInt(1))
    write(keyOutput, valueOutput)
  }
}
```

```
class TopLikesReducer extends ToTableBinaryReducerFx(PersonTable) {
  val totalLikers = values.size
  if (totalLikers > 20) {
    val like = readKey(_.readUTF)
    println((like, totalLikers))
  }
}
```

Demo!!!

Next Steps

- Sample code and slides: blendlabsinc.com
- Challenges
  - Find the influencers -- leaders and followers
  - Find related likes
- Send questions and solved challenges to challenges@blendlabsinc.com
- Talk to us afterwards to help us create a better, more personalized web

Appendix - HBase Example: Email Table

- What operations do we want to be fast?
  - **Inbox:** Get a user's most recent messages in order
  - **Search:** Get headers for all of a user's messages to/from a specific person
  - **Random access:** Get a user's message by timestamp & message-ID

Inbox Table			
row key	rawemail:	header:From	header:Subject
a@a.com#70#message7511	MIME-Version: 1.0\r \n...	z@z.com	Re: Hello!
a@a.com#65#message1771	...	z@z.com	Hello!
b@b.com#68#message9074	...	x@x.com	Fwd: funny video

Search Table	
row key	header:Subject
a@a.com#z@z.com#message7511	Re: Hello!
a@a.com#z@z.com#message1771	Hello!
b@b.com#x@x.com#message9074	Fwd: funny video