Big Data and Functional Programming

Jim Baker

jim.baker@{colorado.edu|python.org|rackspace.com}

Overview

- What is "Big Data"?
- Why is PoPL - a theory course - one of the most pragmatic courses in the CS curriculum?
- A: functional programming
- Explore Big Data and FP, especially by thinking about functions and how they combine
- Specific approaches like MapReduce and the Lambda architecture
- Embracing failure and supporting scale free computation
- FP as the foundation of such companies as GOOG ($357B market cap), FB ($143B), TWTR ($24B)

About me

- Racker working on auto scaling, scalable real time architectures, and OpenStack
- Core developer of Jython
- Co-author of *Definitive Guide to Jython* from Apress
- Occasional teacher of this class (spring 2013 and fall 2013)
- Leader, Boulder/Denver Storm Users meetup
- Formerly, part of original developer team of Ubuntu Juju - lots of experience with ZooKeeper

Internship available

- Still not too late!
- Work on Scala and Big Data problems in Austin this summer
- Strong student in this class
- Talk to me during lunch if you're interested

Lunch plans?

- Lunch today on the *practice of computer science*
    - Meet at `WHEN` in C4C lobby
    - `WHEN` + 0:05 - sit together on the west side of C4C
- So `WHEN` should we meet?

What is "Big Data" anyway?

- Many definitions in common play
- Useful summary paper on the these definitions - "Undefined By Data: A Survey of Big Data Definitions", Jonathan Stuart Ward and Adam Barker, http://arxiv.org/pdf/1309.5821v1.pdf

Gartner

3 Vs of Big Data:

- Volume

- Could equally be true of previous efforts in data warehousing, business intelligence
- Still valid for the business case
- A bit of a leap to a CS definition

Gartner

3 Vs of Big Data:

- Volume
- Velocity

- Could equally be true of previous efforts in data warehousing, business intelligence
- Still valid for the business case
- A bit of a leap to a CS definition

Gartner

3 Vs of Big Data:

- Volume
- Velocity
- Variety

- Could equally be true of previous efforts in data warehousing, business intelligence
- Still valid for the business case
- A bit of a leap to a CS definition

Oracle

Big data combines

- Relational database

Plays with Oracle's preeminence as a proprietary RDBMS vendor

Oracle

Big data combines

- Relational database
- New types of data sources - cited as unstructured, but generally have some structure

Plays with Oracle's preeminence as a proprietary RDBMS vendor

Oracle

Big data combines

- Relational database
- New types of data sources - cited as unstructured, but generally have some structure
- Need new tools to help with this combination

Plays with Oracle's preeminence as a proprietary RDBMS vendor

Intel

- Focus on volume

Intel helpfully sells hardware to process this data

Intel

- Focus on volume
- "Generating a median of 300 terabytes (TB) of data weekly"

Intel helpfully sells hardware to process this data

Microsoft

Big data is the term increasingly used to describe the process of applying serious computing power - the latest in machine learning and artificial intelligence - to seriously massive and often highly complex sets of information.

Ward & Barker

Big data is a term describing the storage and analysis of large and or complex data sets using a series of techniques including, but not limited to: NoSQL, MapReduce and machine learning.

Still puzzled?

- Ward & Barker capture some key ideas
- Still unsatisfying
- Another approach: family of related computational models that support *scale free computing* on data

Scale free computing

- Inherently functional idea, goes back to the original idea of MapReduce
- Intuitive idea: I can run the same program on my laptop as I do on 1000 node compute cluster
- Expect to see (near) linear scale-up in some useful way - size of problem, response time, or both
- Every day evidence of scale free at work - think Google Search

MapReduce

What is this MapReduce idea?

- Map - or sometimes `flatMap`
- Reduce - might also call this `fold`

Map

Related to such ideas as

- Scatter in scatter/gather

Data is consistently mapped to the same node in a given cluster

Map

Related to such ideas as

- Scatter in scatter/gather
- Divide & conquer

Data is consistently mapped to the same node in a given cluster

Map

Related to such ideas as

- Scatter in scatter/gather
- Divide & conquer
- Problem partition

Data is consistently mapped to the same node in a given cluster

Reduce

Word count problem

- What is the word count problem?
- Vs how we usually say it - “wordcount”

Scalding

- High-level domain specific language (DSL) in Scala for writing map-reduce jobs
- Runs on top of Cascalog
- Expect to see more of Scalding in this class
- Or perhaps your future work!

Word count in Scalding


```
import com.twitter.scalding._

class WordCountJob(args : Args) extends Job(args) {
  TextLine( args("input") )
    .flatMap('line -> 'word) {
      line : String => line.split("""\s+""") }
    .groupBy('word) { _.size }
    .write( Tsv( args("output") ) )
}
```

Word count in Summingbird


```
def wordCount(
  source: Iterable[String],
  store: MutableMap[String, Long]) =
source.flatMap {
  sentence =>
  toWords(sentence).map(_ -> 1L)
  }.foreach {
  case (k, v) =>
  store.update(k, store.get(k) + v) }
```

Why is it classic?

- Actually useful
- N grams
- Machine translation of natural language
- Historical usage of words and phrases

Ranking followers with PageRank

```
val sc = new SparkContext(...)
val users = sc
  .textFile("hdfs://user_attributes.tsv")
  .map(line => line.split)
  .map( parts => (parts.head, parts.tail))
val followerGraph = Graph.textFile(sc, ...)
val graph = followerGraph.outerJoinVertices(users){
  case (uid, deg, Some(attrList)) => attrList
  case (uid, deg, None) => Array.empty[String] }
val pagerankGraph = Analytics.pagerank(graph)
val userInfoWithPageRank =
  graph.outerJoinVertices(pagerankGraph.vertices) {
    case (uid, attrList, Some(pr)) => (pr, attrList)
    case (uid, attrList, None) => (pr, attrList)
  }
println(userInfoWithPageRank.top(5))
```

Why functional programming?

- Scala is not just a convenient language to write code
- Could also write such programs in Python or your favorite language
- But something deeper - use how functions combine to support scale free
- Can also rewrite our functions in certain cases - query optimization

Function properties

Some possible properties:

- The object f we call a function is in fact a function!
- Identity - $f(z) = z$
- Associativity (usually)
- Commutativity (very useful when feasible, but hard to reason about)
- Totality (vs partial functions)
- Any other properties?
- Idempotence
- Referential transparency

Referential transparency

- Captures an idea that a given function f is a black box
- But one that's not capturing some state

Referential transparency

Definition: e is *referentially transparent* if we can replace e with its v in **all usages**

Associativity

$$f(f(x, y), z) = f(x, f(y, z))$$

- f is associative if order of operations can be freely *rearranged*
- But does not imply we can freely rearrange the sequence of operands
- Alternative perspective: we can put the parentheses where it makes sense
- What can we parallelize if this holds of our problem?
- What are common examples of nonassociative functions/operations?
- $\Rightarrow$ certainly anything with side effects!

Monoids

- Totality (closed over an operation), associative for operation ("plus"), identity ("zero")
- What does this look like?
- $\Rightarrow$ folds!
- Remember the definition of `foldLeft`

Twitter's Algebird

- Blog post announcing Algebird
- Algebird source
- Fantastic perspective - Programming isn't Math
- Foundation of Summingbird, Scalding

Monads

- Monads are monoids in the category of endofunctors...
- Most monads we have seen, we are interested in sequencing composable operations, taking advantage of associativity
- Back to this later! Let's explore one interesting detail...

At scale, sequencing is expensive!

- Local sequencing is fairly cheap
- Maintaining order requires communication
- Communication proceeds no faster than the speed of light
- Unless we have ansibles ;)

Question

How far does light in a vacuum approximately travel in one **nanosecond**?

- A - 1 kilometer

Question

How far does light in a vacuum approximately travel in one **nanosecond**?

- A - 1 kilometer
- B - 1 meter

Question

How far does light in a vacuum approximately travel in one **nanosecond**?

- A - 1 kilometer
- B - 1 meter
- C - 1 foot

Question

How far does light in a vacuum approximately travel in one **nanosecond**?

- A - 1 kilometer
- B - 1 meter
- C - 1 foot
- D - 1 cm

Question

How far does light in a vacuum approximately travel in one **nanosecond**?

- A - 1 kilometer
- B - 1 meter
- C - 1 foot
- D - 1 cm
- E - 1 mm

An interesting unit: light-foot

- Useful unit: a *light-foot* $\approx$ 1.0167 nanoseconds
- Useful in the same way that units like tablespoons are useful - everyday intuitions
- Pioneering computer scientist Grace Hopper liked to talk about this unit
- Need to consider the **velocity factor**
- Consider a 1 foot USB cable:
  - No specifics about velocity factor on USB cables I could find
  - But gives some insight into what a nanosecond really is

Data center design

- It's all about the locality, to minimize communication hops and distance
- Same core, same chip, same board, same unit, same rack, same aisle, same data center...
- Design focused on communication latency as much as it's storage, computation

Data centers, illustrated

- Google streetview in the datacenter

Multidata center coordination

- Big problem because of communication bottlenecks
- Bigger problem because of data center connection reliability
- These issues are **related**!
- Datacenters are now distributed around the world
- Observations of ping time between cities by one network provider
- What could possibly go wrong?!!

Special relativity and programming

- Good intro/reminder of special relativity
- Einstein and clocks in Bern, Switzerland in 1905
- Einstein was pondering the implication of two things:
  - Established principle since Galieo of the relativity of frames of reference
  - The speed of light (c) is constant, as established by the then recent Michelson-Morley experiment

Einstein's analysis

- Einstein showed the relativity of simultaneity
- We cannot make statements about the whether two spatially-separated events are truly simultaneous
- We can only say something about causality - that A depends on B, which depends on C

Causality and coordination

- Agreement protocols depend on exchanging messages
- Could explore this in a future CS course - Paxos and related coordination protocols
- Support such ideas as *leader election*, *distributed transactions*, and *distributed counters*
- Can we avoid coordination costs?
- $\Rightarrow$ yes! (at least sometimes)

Commutativity

$$f(x, y) = f(y, x)$$

- f is commutative if order of operands can be freely *rearranged*
- What can we parallelize if this holds of our problem?
- What are common examples of noncommutative functions/operations?

Eventual consistency

- Operations can be re-ordered
- Example: Amazon shopping cart
- Databases includign Cassandra, Riak, Dynamo

Shopping cart, with commutative operations

...

Restoring strong consistency across data centers

- Start with high precision time
- Local atomic clocks for *precision*, GPS for *accuracy*
- Network Time Protocol $\rightarrow$ Precision Time Protocol for submicrosecond accuracy
- Ready availability of relatively inexpensive atomic clocks
- Add time versioned databases
- Google's Spanner database implements this idea
- Q: how does this avoid violating the relativity of simultaneity?