Just Enough Math

OSCON tutorial by
Paco Nathan @pacoid

co-author
Allen Day @allenday

http://justenoughmath.com/

Section 2: Abstractions, Isolations, and Lines

Abstract Algebra

Abstract Algebra: TL;DR

Enterprise Data Workflows… *functional programming mitigates software engineering costs for interdisciplinary teams working on cluster computing*:

- abstract algebra allows compilers to compute algebra about functions
- greatly reduced latency, e.g., for real-time analytics or streaming application
- avoids the bottlenecks one typically encounters in batch processing at scale
- exponential cost savings

Abstract Algebra

Show Me The Monoid

Abstract Algebra:

Math papers are concise, but tend to be tough to read… Here's an example of a *good* one:

"Introduction to Semigroups and Monoids"
Peter L. Clark @UGA
http://math.uga.edu/~pete/semigroup.pdf

A **group** is a monoid M in which each element has an inverse.[3]

Exercise 2.2: a) Show that a monoid M is a group iff: for each $x \in M$, the maps

$$x\bullet : M \to M,\ y \mapsto xy, \quad \bullet x : M \to M,\ y \mapsto yx$$

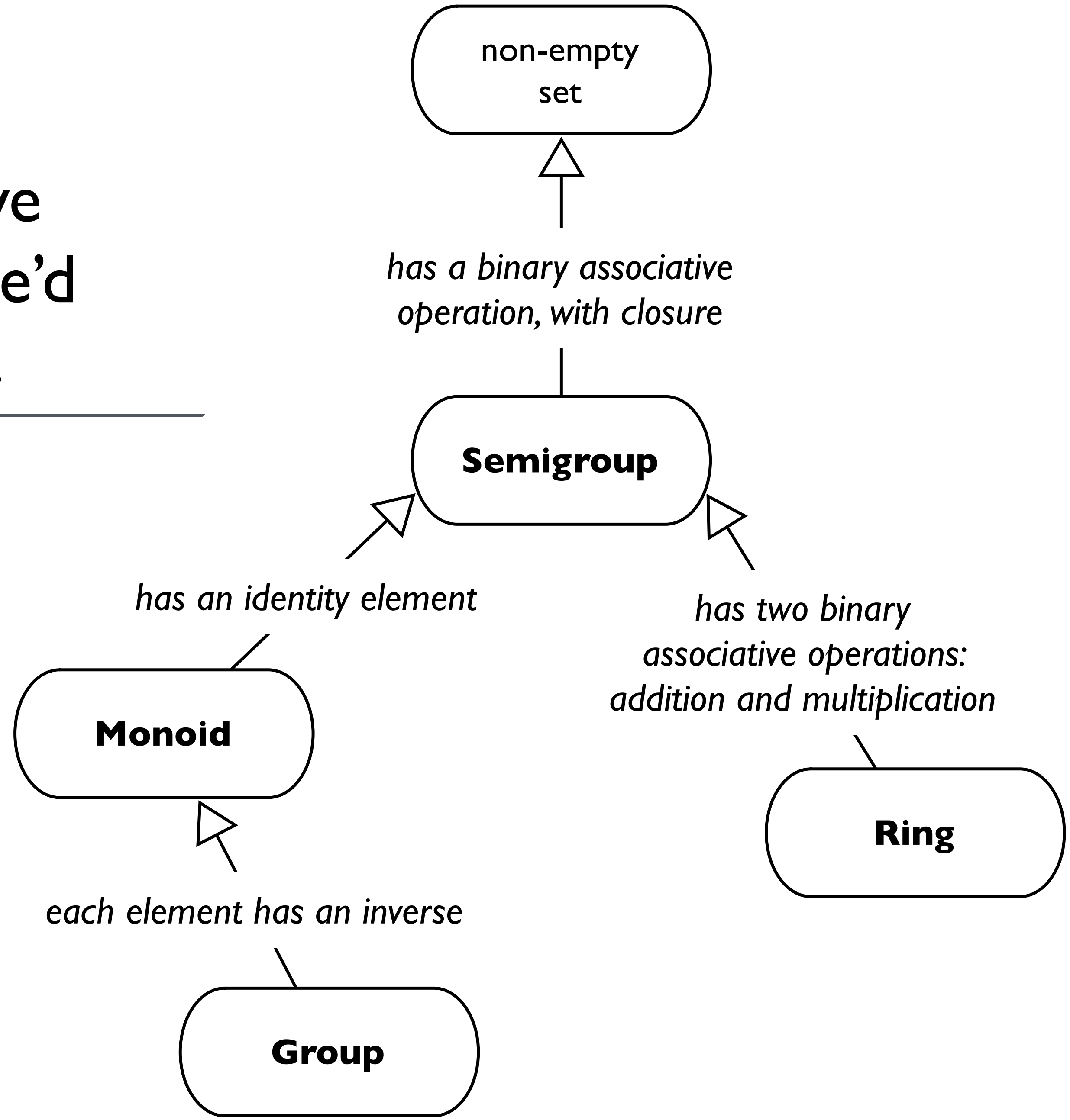
are both bijections.
b) A nontrivial group has no absorbing element.
c) For any monoid M, neither M^e nor M^a is a group.

Exercise 2.3: Show that any group G is isomorphic to its opposite group M^{op}.

The subclass of groups is in many ways simpler and better behaved than the class of all monoids. In this section we explore the following theme: suppose M is a monoid which is not a group: what can we do about it?

Abstract Algebra:

Instead, here is
a cheat-sheet we
really wished we'd
had in school…



Abstract Algebra:

A *semigroup* is a non-empty set with an associative binary operation. For example, addition of integers:

$$(2+3)+4 = 2+(3+4) = 9$$

A *monoid* is a semigroup with an *identity element*:

$$2+0=2$$

That may seem trivial … until you need to *aggregate* billions of complex objects, especially with real-time requirements

course theme! *aggregation*

Abstract Algebra

Functional Programming

Functional Programming:

Theory, Eight Decades Ago:

Haskell Curry, known for seminal work on *combinatory logic* (1927)

Alonzo Church, known for *lambda calculus* (1936) and much more!

Both sought formal answers to the question, *"What can be computed?"*

Alonso Church
wikipedia.org

Haskell Curry
haskell.org

Functional Programming:

Praxis, Four Decades Ago:

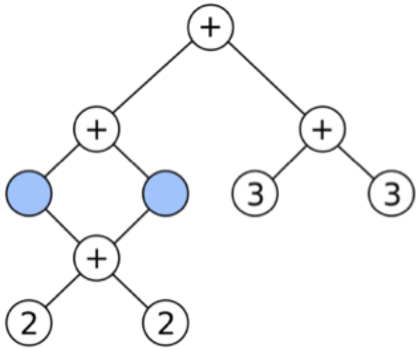
Leveraging lambda calculus, combinators, etc., to increase *parallelism* of apps as applicative systems

John Backus
acm.org

"Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs"
ACM Turing Award (1977)
stanford.edu/class/cs242/readings/backus.pdf

"A new implementation technique for applicative languages"
Turner, D.A. (1979)
Softw: Pract. Exper., 9: 31–49. doi: **10.1002/spe.4380090105**

David Turner
wikipedia.org

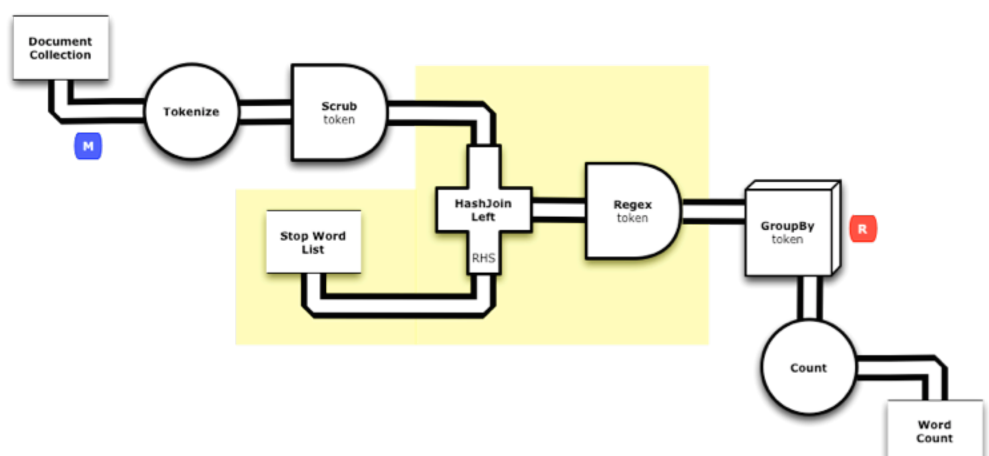


course theme! *parallelism*

Functional Programming:

Chris Wensel created a Java API called *Cascading* (2007) as a way of building *Enterprise data workflows* atop Hadoop

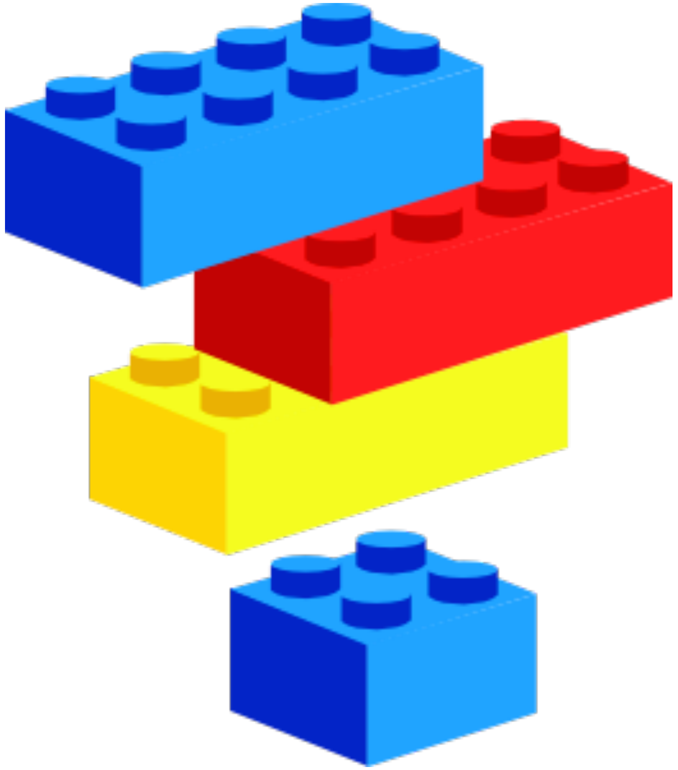
Rather than code directly in MapReduce, developers use some aspects of functional programming inside Java to define data workflows



Functional Programming:

Workflows definitions in Cascading use *function composition* to build pipelines, much the same as in Algebra 2…

$$v = g(y) = g(f(x))$$
$$= 2(x+3)+1$$



key point: *intermediate values* flowing through those pipelines are well-defined and fit together nicely based on the math, using a computable *schema*

course theme! *parallelism*

Functional Programming:

Twitter introduced *Scalding* (2012) as a Scala API for Cascading…

Scala is a descendent of *Haskell*, and arguably much more popular

Twitter reworked their rev apps based on Scalding and have been evangelizing its adoption: now used at scale by eBay, LinkedIn, etc.

engineering.twitter.com/opensource/projects/scalding

Computational Thinking

Decomposition:
Using an "algebra of functions" we can compute about programs, aka *code as data*

➡ *this allows compilers to become much more powerful for parallel processing*

Especially when that code may be applied to a wide variety of data objects, ranging from numbers to customer profiles to matrices, etc.

Computational Thinking

Pattern Recognition:
By leveraging semigroup structure, we can build code libraries that operate on a wide variety of structured data and system architecture

➡ *greater code reuse, less code to test/debug*

Many developers working with Big Data use semigroups in their code already, probably without realizing it – or worse, without being able to describe it

Computational Thinking

Algorithm Design:
Monoids allow for efficient partitioning of the data and required computational tasks across a cluster

➡ *then we can split a wide range of computing programs (graphs, matrices, etc.) into little chunks, yet reassemble the parts at the end*

Abstract Algebra

Performance Bottlenecks

Performance Bottlenecks:

Add ALL the Things:
Abstract Algebra Meets Analytics
infoq.com/presentations/abstract-algebra-analytics
Avi Bryant, Strange Loop (2013)

Avi Bryant
@avibryant

- *grouping doesn't matter (associativity)*
- *ordering doesn't matter (commutativity)*
- *zeros get ignored*

In other words, while partitioning data at scale is quite difficult, you can let the math allow your code to be flexible at scale

course theme! *aggregation*

Performance Bottlenecks:

Algebra for Analytics
speakerdeck.com/johnynek/algebra-for-analytics
Oscar Boykin, Strata SC (2014)

- *"Associativity allows parallelism in reducing"* by letting you put the () where you want
- *"Lack of associativity increases latency exponentially"*

Oscar Boykin
@posco

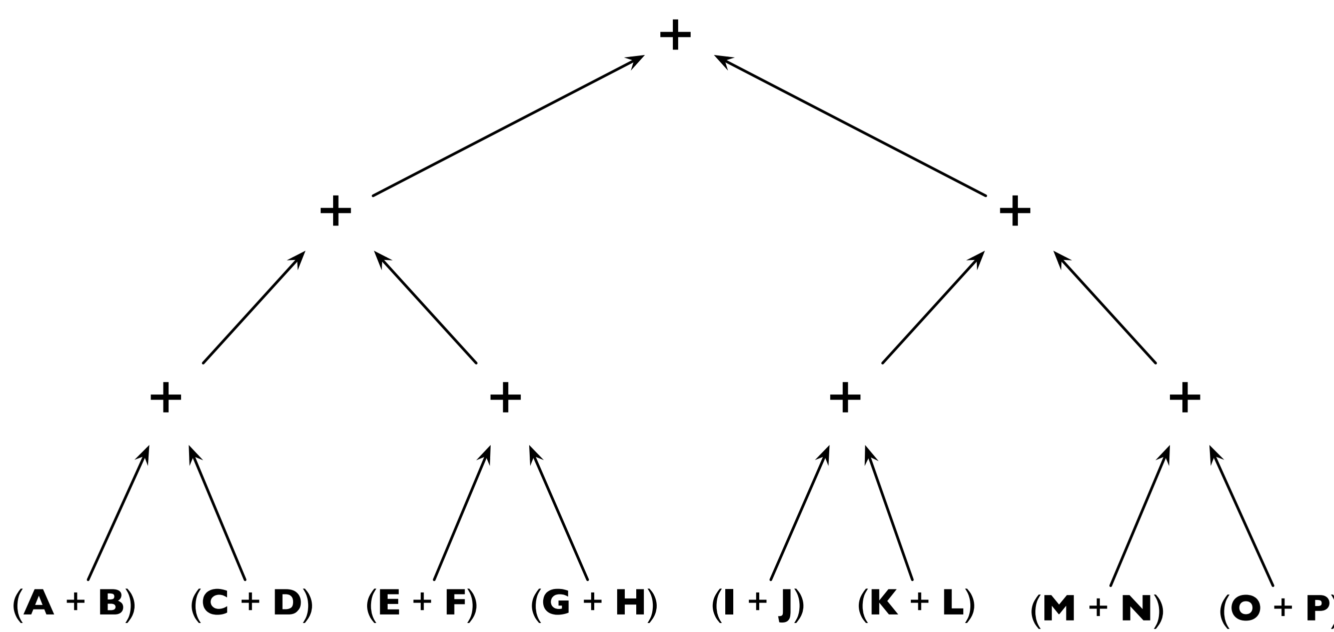
course theme! *parallelism*

Performance Bottlenecks:

Algebra for Analytics
Oscar Boykin, Strata SC (2014)

A + **B** + **C** + **D** + **E** + **F** + **G** + **H** + **I** + **J** + **K** + **L** + **M** + **N** + **O** + **P**

(**A** + **B**)
+ **C**
+ **D**
+ **E**
+ **F**
+ **G**
+ **H**
+ **I**
+ **J**
+ **K**
+ **L**
+ **M**
+ **N**
+ **O**
+ **P**



latency = (N - 1) = 15

latency = log2(N) = 4

Computational Thinking

Algorithm Design:
Guarantees of associativity within the code allow for distributed computing: partial aggregates, tree/graph representation, etc.

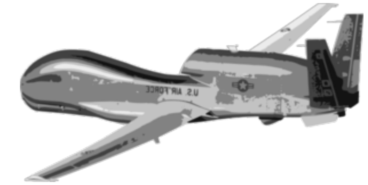
➡ *less resources need to be spent on sorting and windowing data prior to working with a data set*

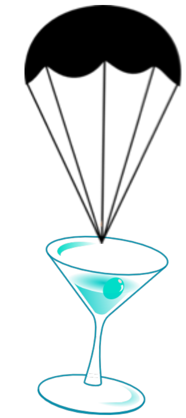
➡ *real-time apps, which don't have the luxury of anticipating data partitions, can respond quickly*

Abstract Algebra

Monoids in Python

Monoids in Python:





Foobartendr.io has a real-time app that streams through web logs, looking for patterns of text where customers have been impressed, dissatisfied, etc.

Problem: *write monoids in Python that add lists and dictionaries, to avoid potential bottlenecks in the streaming analytics*

Monoids in Python:

To get started on this, first check out an excellent discussion about writing monoids in Python:

"Monoids in Python"
Francisco Mota (2011)

fmota.eu/blog/monoids-in-python.html

Monoids in Python:

Key points about monoids in Python:

function	*effect*
`lift()`	like a *map* in MapReduce
`op()`	like a *reduce* in MapReduce
`fold()`	applies the monoid to a list of values
`star()`	applies the monoid to a list of lists of values

Monoids in Python:

Go to your browser window at:

http://localhost:8888

Double-click on:

lesson_01_show_me_the_monoid

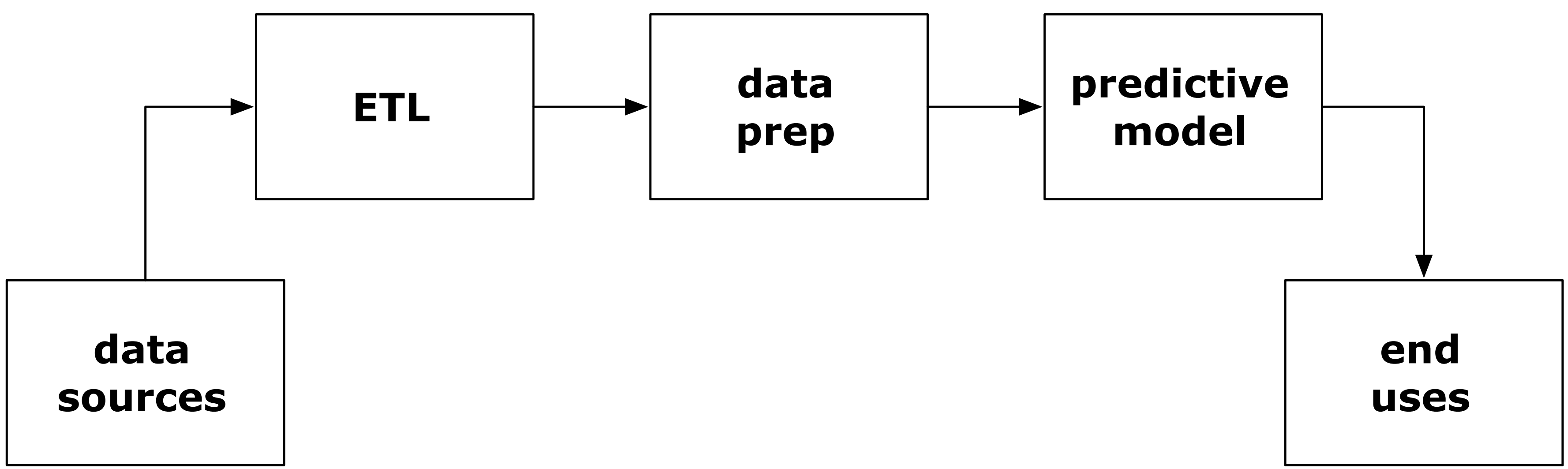
Follow the instructions for each programming exercise

Abstract Algebra

Data Workflows

Data Workflows:

Effectively, *middleware* is evolving for Big Data and Machine Learning… the following design pattern (a DAG) shows up in many places…



Data Workflows:

Effectively,
Machine L ttern
(a DAG) s

workflows are inherently about both automation *and* people:

- separation of concerns
- literate programming
- visualized business process
- leveraging pattern language
- operationalizing solutions

Data Workflows:

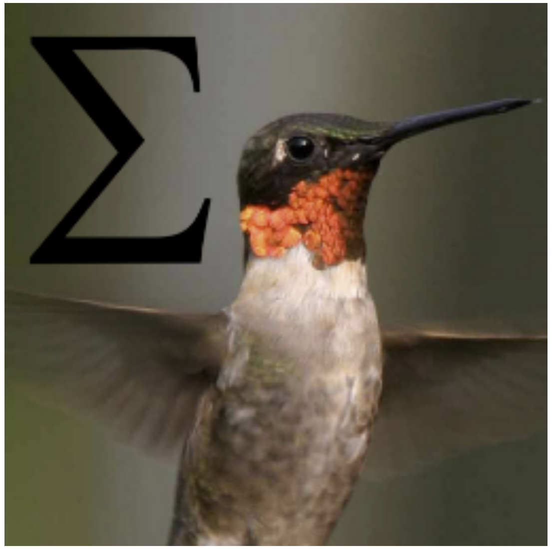
Fortunately, *workflow abstraction layers* have been evolving, as a kind of Big Data middleware…

slideshare.net/pacoid/data-workflows-for-machine-learning

Data Workflows:

Twitter released an open source Scala library called *Algebird* in 2012 that provides abstract algebra definitions for Scalding, Spark, etc.

engineering.twitter.com/opensource/projects/algebird



Data Workflows:

Oscar Boykin, co-author of *Scalding*, *Algebird*, *etc.*, explained on **StackOverflow**:

> *The main answer is that by exploiting semi-group structure, we can build systems that parallelize correctly without knowing the underlying operation (the user is promising associativity).*
>
> *By using Monoids, we can take advantage of sparsity (we deal with a lot of sparse matrices, where almost all values are a zero in some Monoid).*
>
> *By using Rings, we can do matrix multiplication over things other than numbers (which on occasion we have done).*

Oscar Boykin, on **StackOverflow**: (cont'd)

The algebird project itself (as well as the issue history) pretty clearly explains what is going on here: we are building a lot of algorithms for aggregation of large data sets, and leveraging the structure of the operations gives us a win on the systems side (which is usually the pain point when trying to productionize algorithms on 1000s of nodes).

Solve the systems problems once for any Semigroup/Monoid/Group/Ring, and then you can plug in any algorithm without having to think about Memcache, Hadoop, Storm, etc...

course theme! *parallelism*

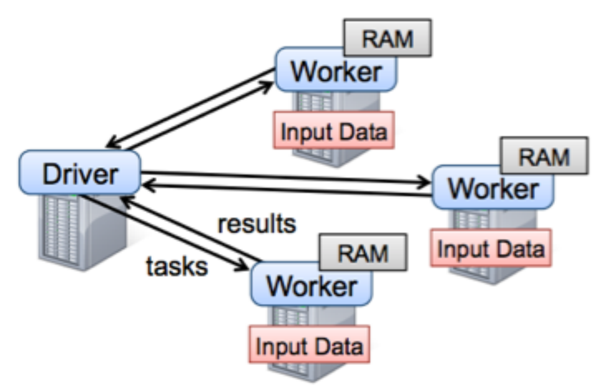
Data Workflows:

Spark: a unified platform for big data analytics: batch, streaming, interactive, graph, ML, SQL, etc.

The State of Spark, and Where We're Going Next
Matei Zaharia
Spark Summit (2013)
youtu.be/nU6vO2EJAb4

Spark SQL: Manipulating Structured Data Using Spark
Michael Armbrust, **Reynold Xin**
databricks.com/blog/2014/03/26/Spark-SQL-manipulating-structured-data-using-Spark.html

spark.apache.org/



Probabilistic Data Structures

Probabilistic Data Structures:

fascinating and relatively new area, pioneered by a few people – e.g., **Philippe Flajolet**

Twitter catch-phrase is: "*Hash, don't sample*"

provides *approximation*, with error bounds – and generally uses significantly less resources (RAM, CPU, etc.)

many algorithms can be constructed from combinations of read and write *monoids*

aggregate different ranges by composing the hashes, instead of repeating full-queries

course theme! *approximation*

Probabilistic Data Structures:

some examples:

algorithm	*use case*
Count-Min Sketch	frequency summaries
HyperLogLog	set cardinality
Bloom Filter	set membership
MinHash	set similarity
DSQ	streaming quantiles
SkipList	ordered sequence search

Probabilistic Data Structures:

some use cases:

- streaming algorithms for real-time analytics, e.g., **quantiles**
- identifying "heavy-hitters" in very large data
- **BlinkDB**: queries with bounded errors and bounded response time on very large data
- **genomics**
- **compressed sensing**

Probabilistic Data Structures:

recommended reading:

Probabilistic Data Structures for Web Analytics and Data Mining
Ilya Katsov

A collection of links for streaming algorithms and data structures
Debasish Ghosh

Aggregate Knowledge blog
Timon Karnezos, Matt Curcio, et al.

Probabilistic Data Structures:

Go to your browser window at:

http://localhost:8888

Double-click on:

lesson_02_prob_data_struct

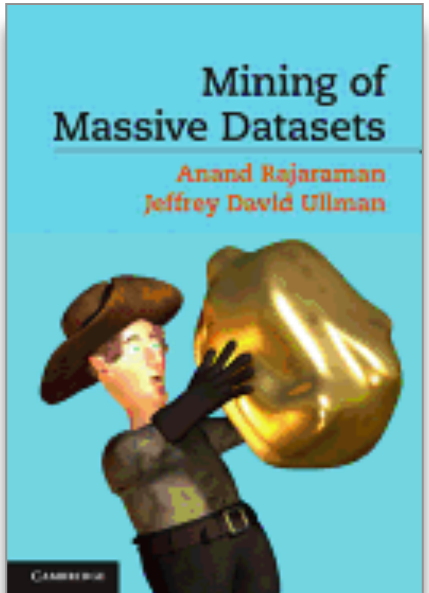
Follow the instructions for each programming exercise

Computational Thinking

Pattern Recognition:
With many large-scale analytics use cases, you'll be approximating anyway… You may be able to leverage approximation algorithms to trade bounded errors for orders of magnitude less required resources

➡ *greatly reduced resources, generally with much better parallelism*

Sidebar: *Recommended Reading*



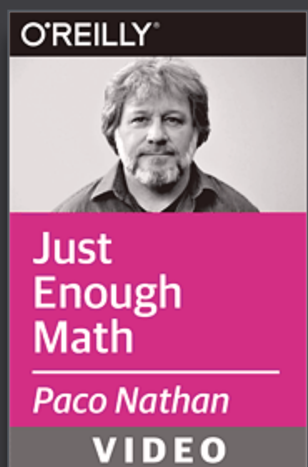
Mining of Massive Datasets
Jure Leskovec,
Anand Rajaraman,
Jeff Ullman
Cambridge (2011)
mmds.org/#book

Algebird
Avi Bryant,
Oscar Boykin, et al.
Twitter (2012)
engineering.twitter.com/
opensource/projects/algebird

(follow-ups)

monthly newsletter for updates, events, conf summaries, etc.:

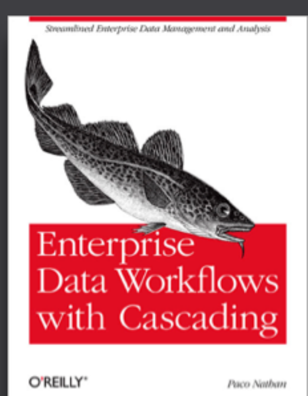
liber118.com/pxn/



Just Enough Math

O'Reilly, 2014

oreilly.com/go/enough_math/
preview: youtu.be/TQ58cWgdCpA



Enterprise Data Workflows with Cascading

O'Reilly, 2013

shop.oreilly.com/product/0636920028536.do

calendar:

Scala by the Bay
SF, Aug 8
scalabythebay.org

#MesosCon
Chicago, Aug 21
events.linuxfoundation.org/events/mesoscon

Cassandra Summit
SF, Sep 10
cvent.com/events/cassandra-summit-2014

Strata NYC + Hadoop World
NYC, Oct 15
strataconf.com/stratany2014

Strata EU
Barcelona, Nov 20
strataconf.com/strataeu2014

Data Day Texas
Austin, Jan 10
datadaytexas.com

many thanks:

Andrew Odewahn & team

Mike Loukides, Ann Spencer, Ben Lorica

Allen Norren, Kirk Walter,
Audrius Reskevicius, Keith Thompson

Yasmina Greco, Kathy Yu, Annalis Clint,
Marsee Henon

Shirley Bailes, Sonia Zapien,
Betsy Waliszewski

Allen Day, Lynn Bender, Chris Severs,
Josh Neland, and many technical
reviewers!