Map Reduce and MRJob

Verena Kaynig-Fittkau

vkaynig@seas.harvard.edu

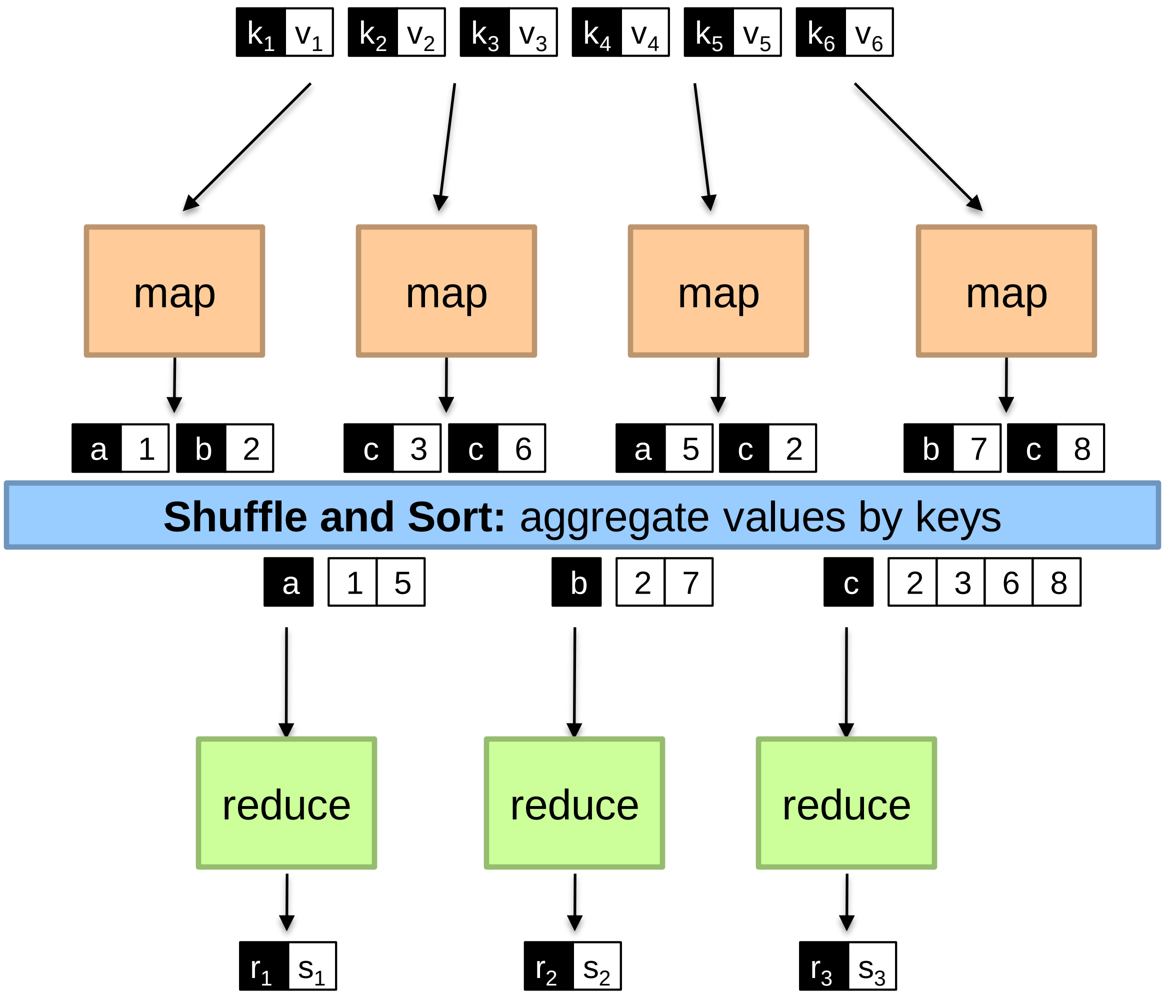
Today's Agenda

- Some Python basics
  - Classes, generators
  - Introduction to MRJob
- MapReduce design patterns

Data-Intensive Text Processing with MapReduce
Jimmy Lin and Chris Dyer.
Morgan & Claypool Publishers, 2010.

MapReduce

- Programming model for distributed computations
- Software framework for clusters
- Massive data processing
- No hassle with low level programming
  - Partitioning input data
  - Scheduling execution
  - Handling failures
  - Intermachine communication
- Open source implementation
- MRJob: Python class for Hadoop Streaming



k_1 v_1 k_2 v_2 k_3 v_3 k_4 v_4 k_5 v_5 k_6 v_6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 3 6 8
reduce
reduce
reduce
r_1 s_1
r_2 s_2
r_3 s_3

The Famous Word Count Example

```
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

Python Classes

```
class Box(object):
    def __init__(self):
        self.nrObjects = 0

    def isEmpty(self):
        return self.nrObjects==0

    def putObjectsIn(self, n):
        self.nrObjects += n

    def printNrOfObjects(self):
        print "Box contains " + \
        str(self.nrObjects) + " objects"
```

Python Classes

```
In [2]: boxA = Box()

In [3]: boxA.
boxA.isEmpty              boxA.printNrOfObjects
boxA.nrObjects            boxA.putObjectsIn

In [3]: boxA.isEmpty()
Out[3]: True

In [4]: boxA.putObjectsIn(5)

In [5]: boxA.printNrOfObjects()
Box contains 5 objects

In [6]: boxB = Box()

In [7]: boxB.putObjectsIn(10)

In [8]: boxB.printNrOfObjects()
Box contains 10 objects

In [9]: boxA.printNrOfObjects()
Box contains 5 objects
```

Derived Class in Python

```
class LabeledBox(Box):
    def __init__(self,l):
        super(LabeledBox, self).__init__()
        self.label = l

    def printLabel(self):
        print "Box is labeled: " + self.label
```

```
class Box(object):
    def __init__(self):
        self.nrObjects = 0
```

Derived Class in Python

```
In [14]: boxL = LabeledBox("Stuff")

In [15]: boxA.
boxA.isEmpty            boxA.printNrOfObjects
boxA.nrObjects          boxA.putObjectsIn

In [15]: boxL.
boxL.isEmpty            boxL.nrObjects          boxL.printNrOfObjects
boxL.label              boxL.printLabel         boxL.putObjectsIn

In [15]: boxL.putObjectsIn(7)

In [16]: boxL.printNrOfObjects()
Box contains 7 objects

In [17]: boxL.printLabel()
Box is labeled: Stuff
```

Overwriting a Function

```python
class LabeledBox(Box):
    def __init__(self,l):
        super(LabeledBox, self).__init__()
        self.label = l

    def printLabel(self):
        print "Box is labeled: " + self.label

    def printNrOfObjects(self):
        print "Box labeled " + self.label + \
        " contains " + str(self.nrObjects) + \
        " objects"
```

Overwriting a Function

```
In [23]: boxA.printNrOfObjects()
Box contains 5 objects

In [24]: boxL.printNrOfObjects()
Box labeled Stuff contains 7 objects
```

The Famous Word Count Example

```
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

Python generators

```
# Build and return a list
def firstn(n):
    num, nums = 0, []
    while num < n:
        nums.append(num)
        num += 1
    return nums

sum_of_first_n = sum(firstn(1000000))
```

http://wiki.python.org/moin/Generators

Python generators

```
# a generator that yields items instead of returning a list
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1

sum_of_first_n = sum(firstn(1000000))
```

http://wiki.python.org/moin/Generators

Build in Python Generators

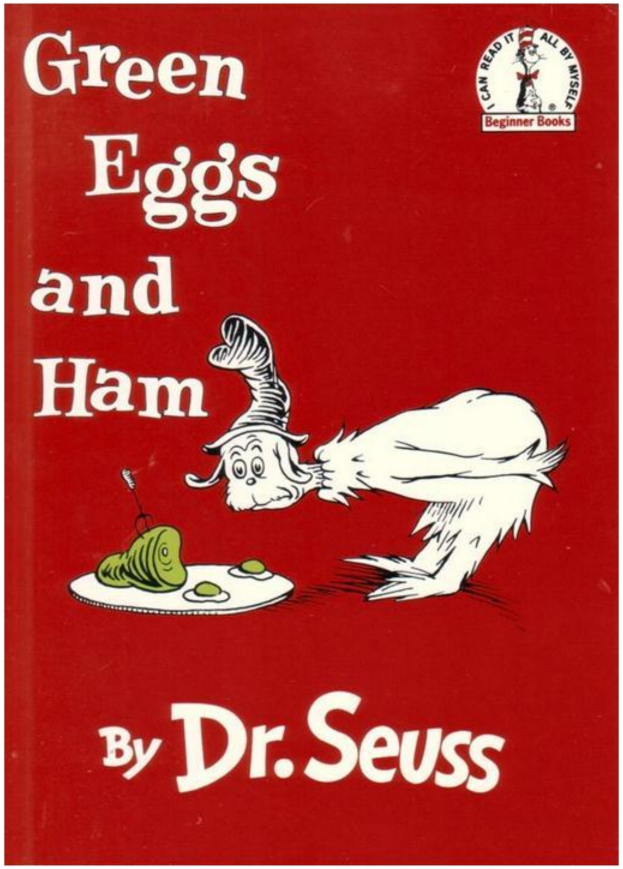
```
# Note: Python 2.x only
# using a non-generator
sum_of_first_n = sum(range(1000000))

# using a generator
sum_of_first_n = sum(xrange(1000000))
```

http://wiki.python.org/moin/Generators

The Famous Word Count Example

```python
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

A sample file



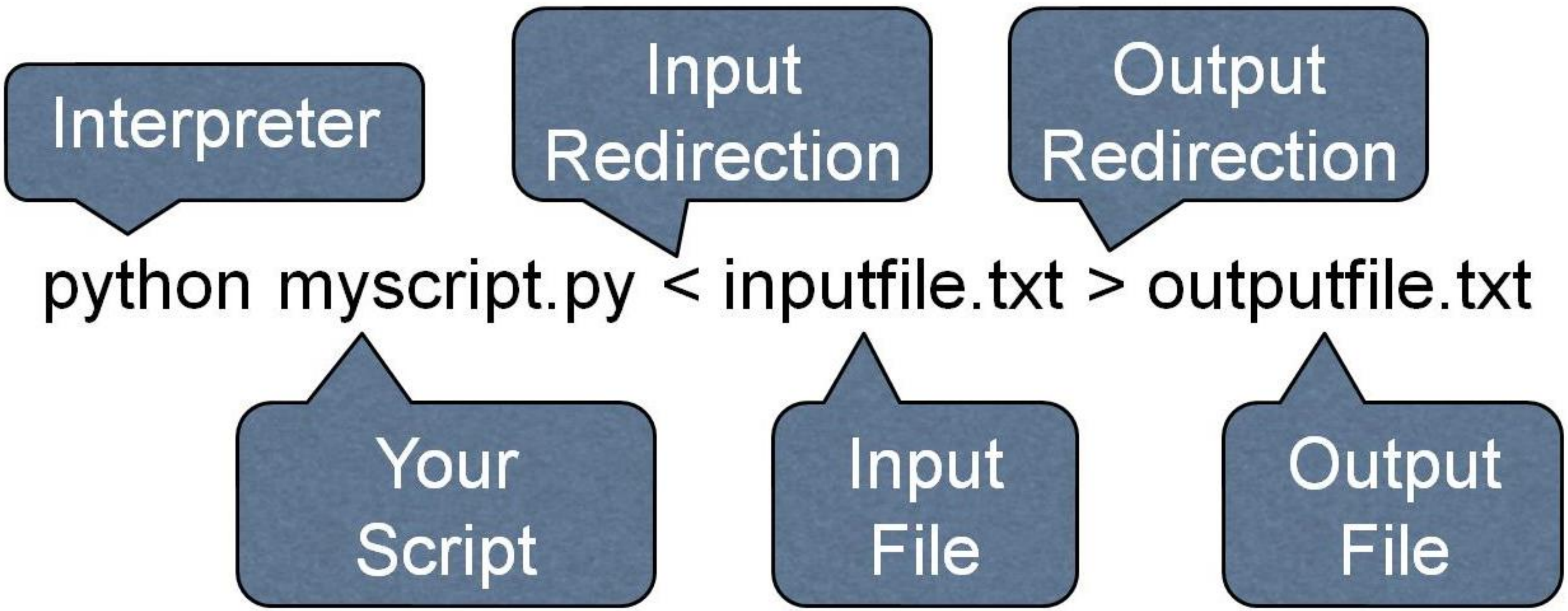
Example Input File

```
I am Sam

I am Sam
Sam I am

That Sam I am
That Sam I am
I do not like
that Sam I am

Do you like
green eggs and ham

I do not like them
Sam I am
I do not like
green eggs and ham
```

```python
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

■ ■ ■

Launching the Job

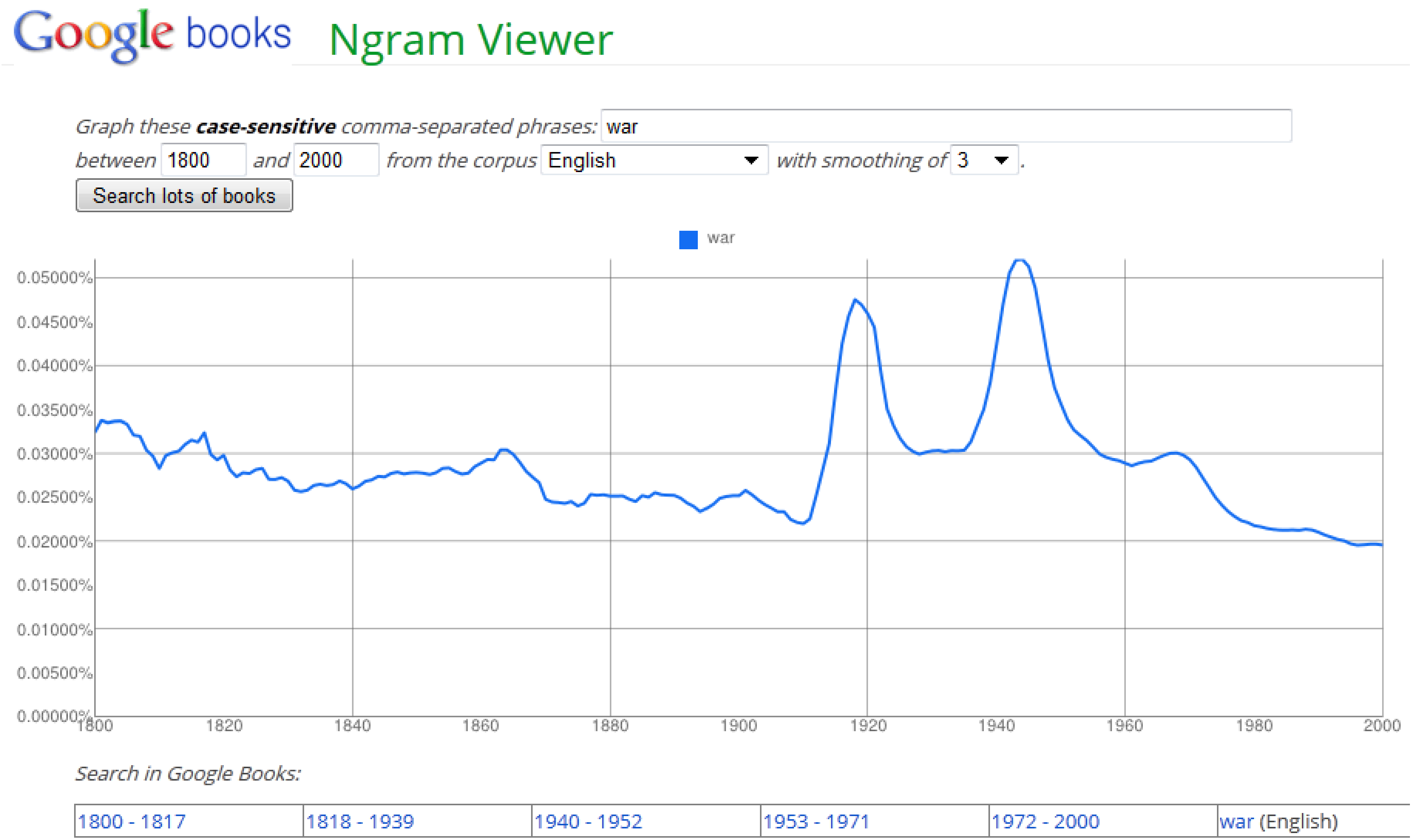


Output File

```
""        36
"a"       60
"am"      16
"and"     24
"anywhere"        8
"are"     2
"be"      4
"boat"    3
"box"     7
"car"     7
"could"   14
"dark"    7
"do"      36
"eat"     24
"eggs"    10
"fox"     7
"goat"    4
"good"    2
"green"   10
"ham"     10
"here"    11
"house"   8
"i"       84
"if"      1
"in"      41
"let"     4
"like"    44
```

50 words in total

Why Counting Words is Fun



http://books.google.com/ngrams

More Culturomics

Google books Ngram Viewer

Graph these ***case-sensitive*** *comma-separated phrases:* Apple

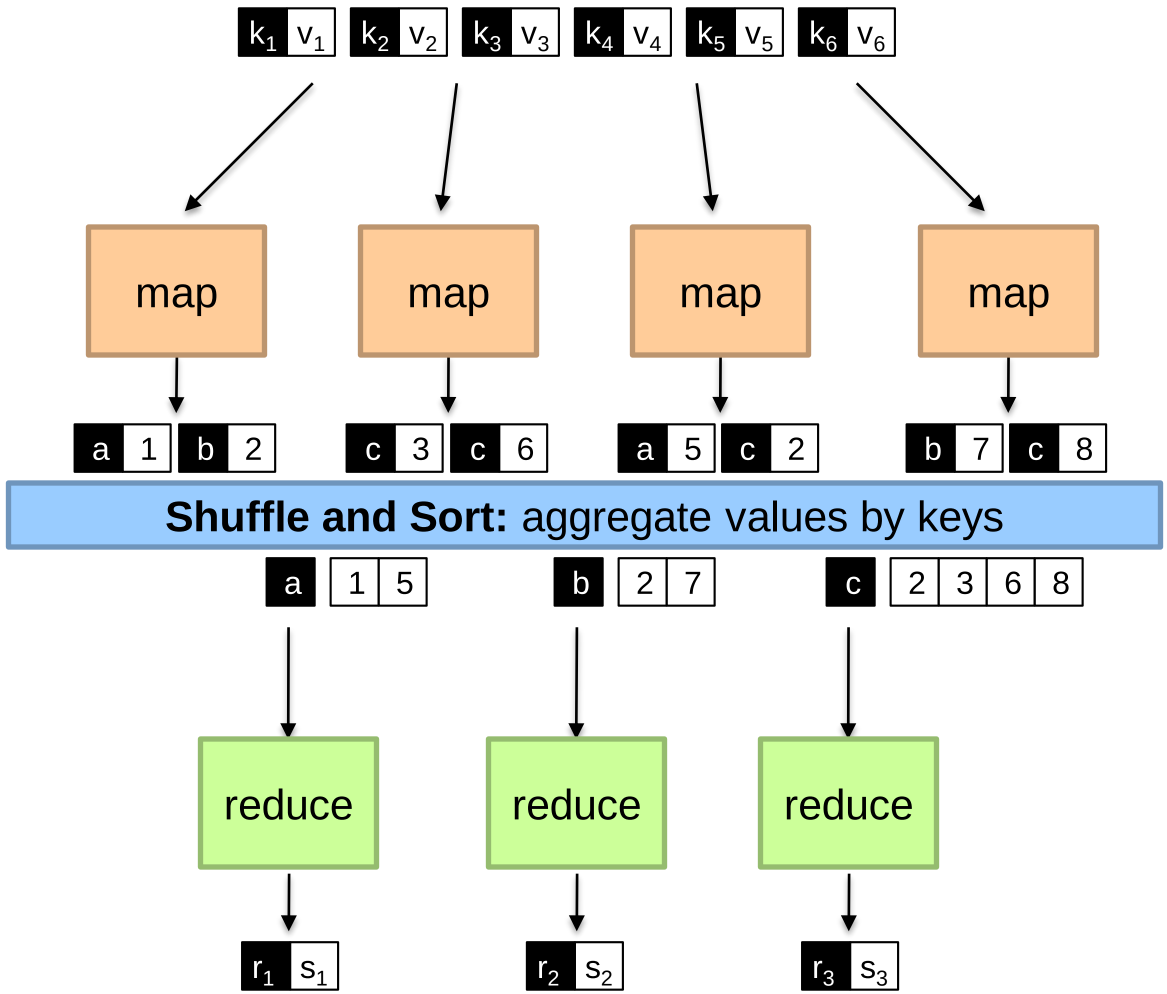
between 1800 *and* 2000 *from the corpus* English *with smoothing of* 3 .

Search lots of books

Search in Google Books:

1800 - 1839	1840 - 1978	1979 - 1987	1988 - 1993	1994 - 2000	Apple (English)

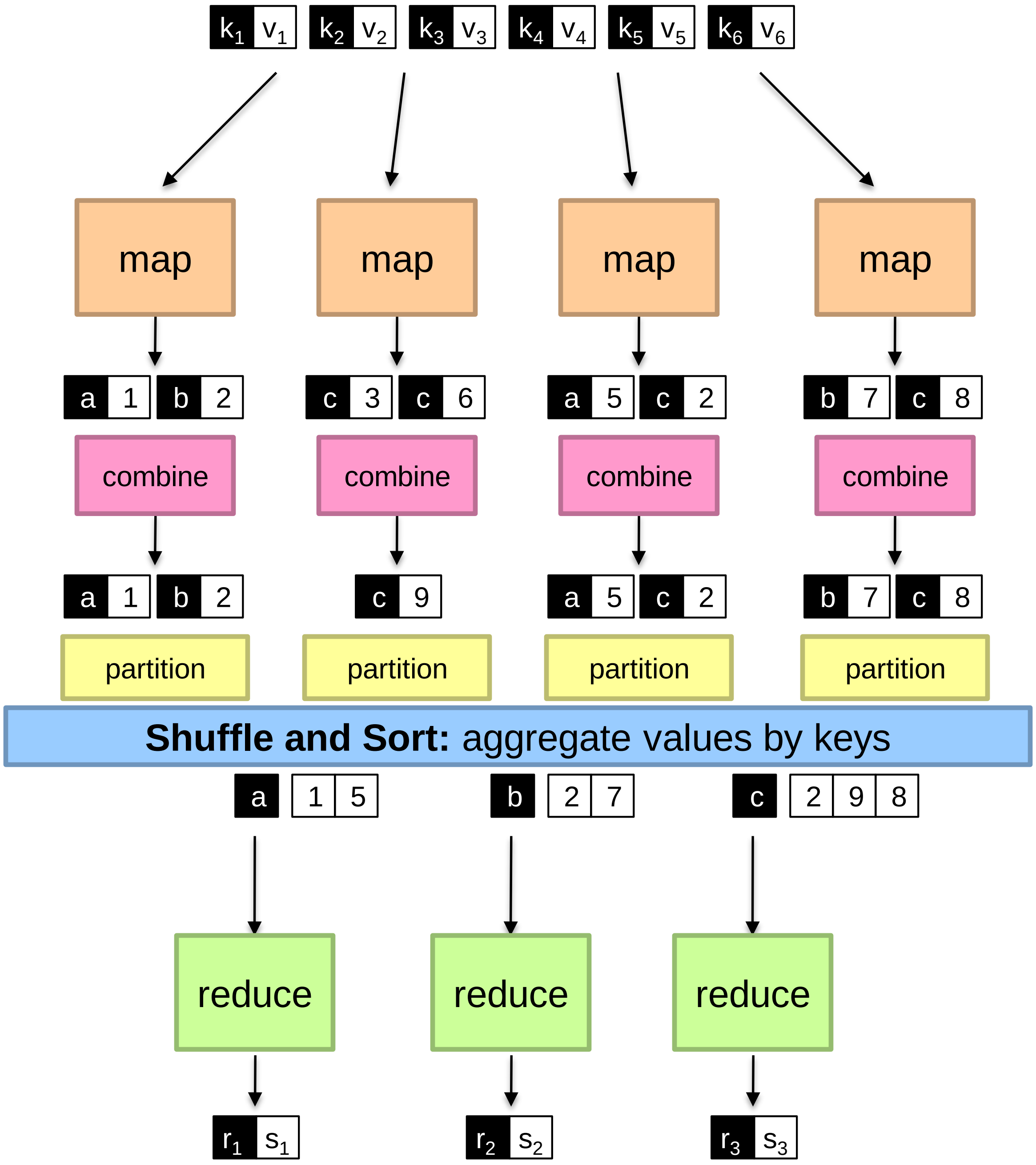
Run your own experiment! Raw data is available for download here.



k_1 v_1 k_2 v_2 k_3 v_3 k_4 v_4 k_5 v_5 k_6 v_6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 3 6 8
reduce
reduce
reduce
r_1 s_1
r_2 s_2
r_3 s_3

Importance of Local Aggregation

- Ideal scaling characteristics:
  - Twice the data, twice the running time
  - Twice the resources, half the running time
- Why can’t we achieve this?
  - Synchronization requires communication
  - Communication kills performance
- Thus… avoid communication!
  - Reduce intermediate data via local aggregation
  - Two possibilities:
    - Combiners
    - In-mapper combining



k1 v1 k2 v2 k3 v3 k4 v4 k5 v5 k6 v6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
combine
combine
combine
combine
a 1 b 2
c 9
a 5 c 2
b 7 c 8
partition
partition
partition
partition
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3

Combiner

- “mini-reducers”
- Takes mapper output before shuffle and sort
- Can significantly reduce network traffic
- No access to other mappers
- Not guaranteed to get all values for a key
- Not guaranteed to run at all!
- Key and value output must match mapper

Word Count with Combiner

```python
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def combiner(self,word,occurrences):
        yield word, sum(occurrences)

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

Combiner Design

- Combiners and reducers share same method signature
  - Sometimes, reducers can serve as combiners
  - Often, not…
- Remember: combiner are optional optimizations
  - Should not affect algorithm correctness
  - May be run 0, 1, or multiple times
- Example: find average of all integers associated with the same key

Computing the Mean: Version 1

class MAPPER
  method MAP(string t, integer r)
    EMIT(string t, integer r)

class REDUCER
  method REDUCE(string t, integers $[r_1, r_2, \ldots]$)
    $sum \leftarrow 0$
    $cnt \leftarrow 0$
    for all integer $r \in$ integers $[r_1, r_2, \ldots]$ **do**
      $sum \leftarrow sum + r$
      $cnt \leftarrow cnt + 1$
    $r_{avg} \leftarrow sum/cnt$
    EMIT(string t, integer r_{avg})

Why can’t we use reducer as combiner?

Computing the Mean: Version 2

class MAPPER
  method MAP(string t, integer r)
    EMIT(string t, integer r)

class COMBINER
  method COMBINE(string t, integers $[r_1, r_2, \ldots]$)
    $sum \leftarrow 0$
    $cnt \leftarrow 0$
    for all integer $r \in$ integers $[r_1, r_2, \ldots]$ **do**
      $sum \leftarrow sum + r$
      $cnt \leftarrow cnt + 1$
    EMIT(string t, pair (sum, cnt)) ▷ Separate sum and count

class REDUCER
  method REDUCE(string t, pairs $[(s_1, c_1), (s_2, c_2) \ldots]$)
    $sum \leftarrow 0$
    $cnt \leftarrow 0$
    for all pair $(s, c) \in$ pairs $[(s_1, c_1), (s_2, c_2) \ldots]$ **do**
      $sum \leftarrow sum + s$
      $cnt \leftarrow cnt + c$
    $r_{avg} \leftarrow sum/cnt$
    EMIT(string t, integer r_{avg})

Why doesn't this work?

Computing the Mean: Version 3

class $\textsc{Mapper}$
  method $\textsc{Map}(\text{string } t, \text{integer } r)$
    $\textsc{Emit}(\text{string } t, \text{pair } (r, 1))$

class $\textsc{Combiner}$
  method $\textsc{Combine}(\text{string } t, \text{pairs } [(s_1, c_1), (s_2, c_2) \ldots])$
    $sum \leftarrow 0$
    $cnt \leftarrow 0$
    for all pair $(s, c) \in \text{pairs } [(s_1, c_1), (s_2, c_2) \ldots]$ **do**
      $sum \leftarrow sum + s$
      $cnt \leftarrow cnt + c$
    $\textsc{Emit}(\text{string } t, \text{pair } (sum, cnt))$

class $\textsc{Reducer}$
  method $\textsc{Reduce}(\text{string } t, \text{pairs } [(s_1, c_1), (s_2, c_2) \ldots])$
    $sum \leftarrow 0$
    $cnt \leftarrow 0$
    for all pair $(s, c) \in \text{pairs } [(s_1, c_1), (s_2, c_2) \ldots]$ **do**
      $sum \leftarrow sum + s$
      $cnt \leftarrow cnt + c$
    $r_{avg} \leftarrow sum/cnt$
    $\textsc{Emit}(\text{string } t, \text{pair } (r_{avg}, cnt))$

Fixed?

In-Mapper Combining

- “Fold the functionality of the combiner into the mapper by preserving state across multiple map calls

```
class Mapper
    method Initialize
        S ← new AssociativeArray
        C ← new AssociativeArray
    method Map(string t, integer r)
        S{t} ← S{t} + r
        C{t} ← C{t} + 1
    method Close
        for all term t ∈ S do
            Emit(term t, pair (S{t}, C{t}))
```

In-Mapper Combining

- Advantages
  - Speed
  - Why is this faster than actual combiners?
- Disadvantages
  - Explicit memory management required
  - Potential for bugs

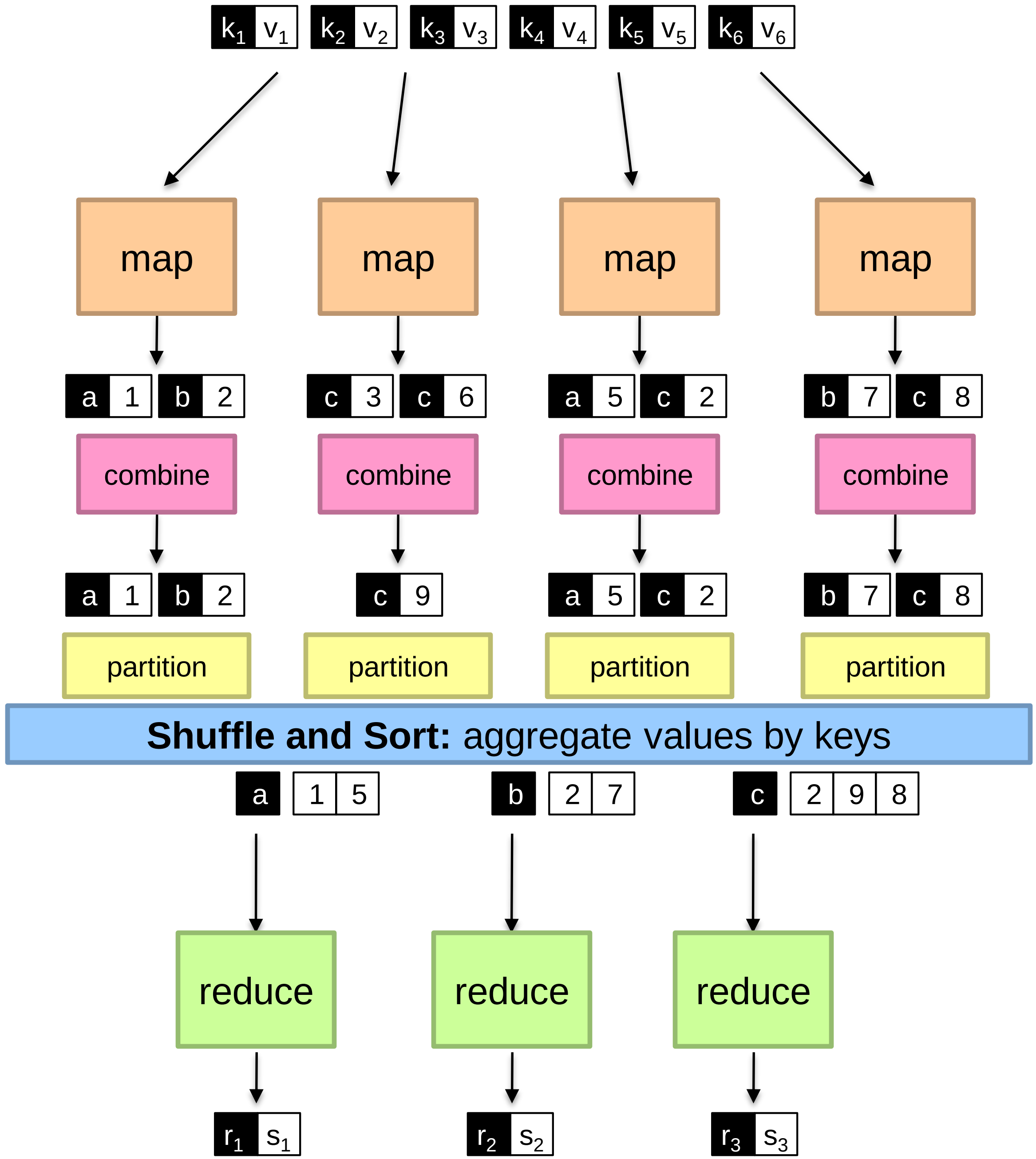
Word Count with In-Mapper-Comb.

```
from collections import defaultdict
from mrjob.job import MRJob

class mrWordCount(MRJob):
    def __init__(self, *args, **kwargs):
        super(mrWordCount, self).__init__(*args, **kwargs)
        self.localWordCount = defaultdict(int)

    def mapper(self,key,line):
        if False:
            yield
        for word in line.split(' '):
            self.localWordCount[word.lower()]+=1

    def mapper_final(self):
        for (word, count) in self.localWordCount.iteritems():
            yield word, count

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

Word of Caution

```python
from mrjob.job import MRJob
import sys

class SimpleTest(MRJob):

    def __init__(self, *args, **kwargs):
        super(SimpleTest, self).__init__(*args, **kwargs)
        self.test = 1

    def mapper(self,key,value):
        self.test = 2
        yield 1, self.test

    def mapper_final(self):
        yield 1, self.test

    def reducer(self, key, value):
        sys.stderr.write(str(self.test))
        yield 1, value

if __name__ == '__main__':
    SimpleTest.run()
```

1!!



k_1 v_1
k_2 v_2
k_3 v_3
k_4 v_4
k_5 v_5
k_6 v_6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
combine
combine
combine
combine
a 1 b 2
c 9
a 5 c 2
b 7 c 8
partition
partition
partition
partition
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
reduce
reduce
reduce
r_1 s_1
r_2 s_2
r_3 s_3

Partitioner

- Decides which reducer gets which key-value pair
- Default depends on hash value of key in raw byte representation
- May result in unequal load balancing
- Custom partitioner often wise for complex keys

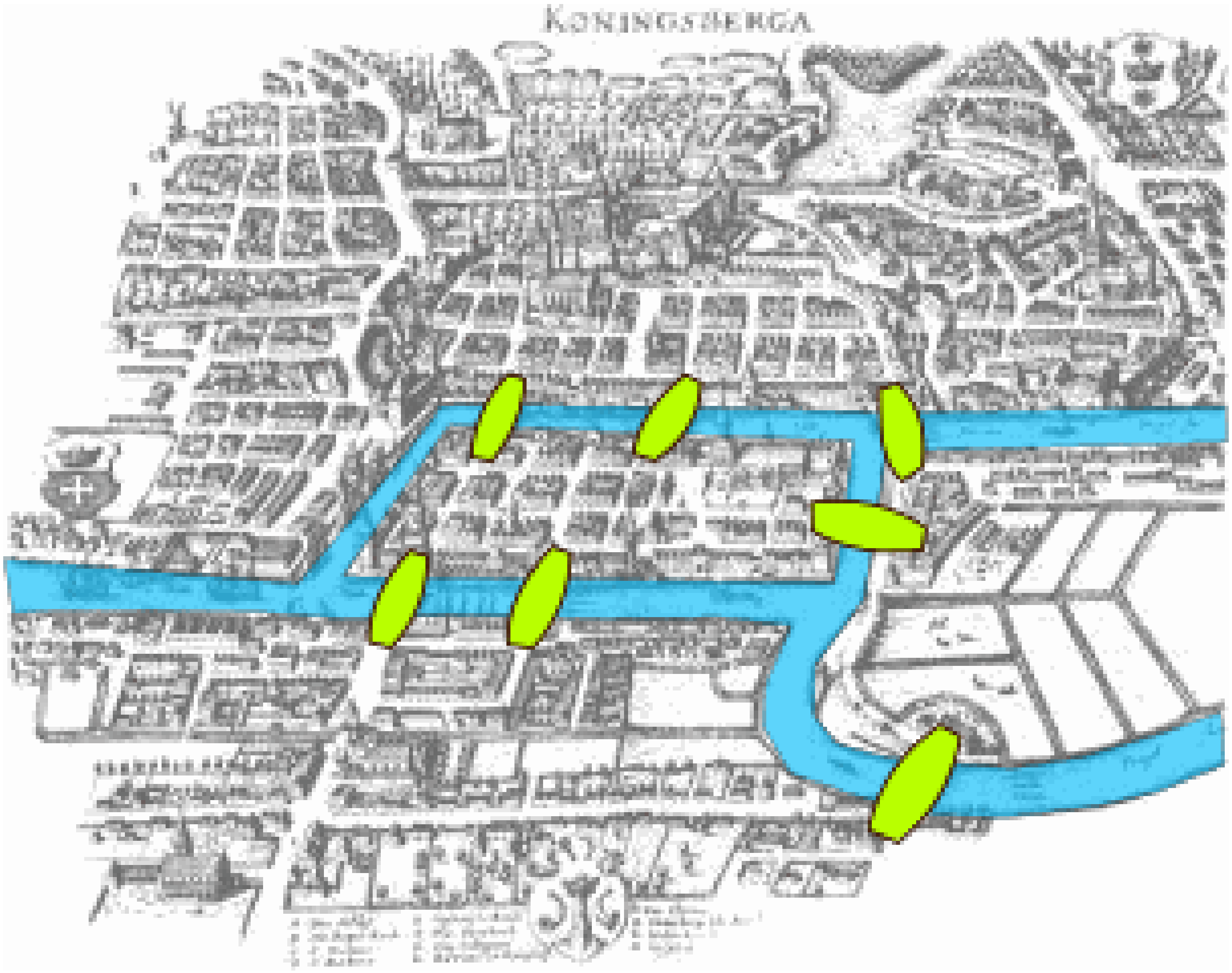
Python Code for Partitioner

- Hadoop has a selection of partitioners
- You can specify which partitioner MRJob should choose:

```
# Partitioner configuration
self.options.partitioner = 'org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner'
self.options.jobconf['mapred.text.key.partitioner.options'] = '-k1,1n'
self.options.jobconf['map.output.key.field.separator'] = ','
```

- The key specification is of the form
  - -kpos1[,pos2]
  - Pos1 is the number of the key field to use
  - Fields are numbered starting with 1
  - The n specifies integer keys

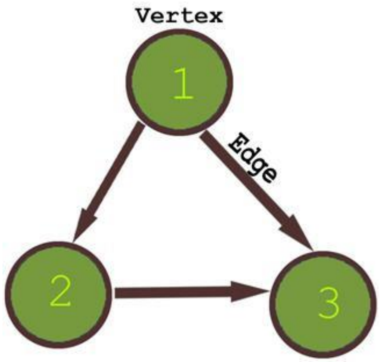
http://hadoop.apache.org/docs/r0.20.1/api/org/apache/hadoop/mapred/lib/KeyFieldBasedPartitioner.html



http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

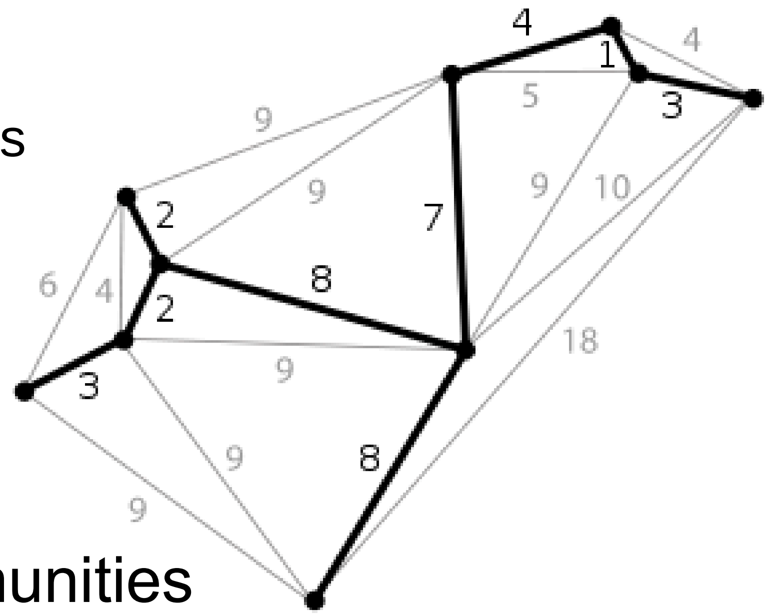
What's a graph?

- G = (V,E), where
  - V represents the set of vertices (nodes)
  - E represents the set of edges (links)
  - Both vertices and edges may contain additional information
- Different types of graphs:
  - Directed vs. undirected edges
  - Presence or absence of cycles
- Graphs are everywhere:
  - Hyperlink structure of the Web
  - Physical structure of computers on the Internet
  - Interstate highway system
  - Social networks



Some Graph Problems

- Finding shortest paths
  - Routing Internet traffic and UPS trucks
- Finding minimum spanning trees
  - Telco laying down fiber



- Identify "special" nodes and communities
  - Breaking up terrorist cells, spread of avian flu
- Bipartite matching
  - Monster.com, Match.com
- PageRank

Graphs and MapReduce

- Graph algorithms typically involve:
  - Performing computations at each node: based on node features, edge features, and local link structure
  - Propagating computations: “traversing” the graph
- Key questions:
  - How do you represent graph data in MapReduce?
  - How do you traverse a graph in MapReduce?

Representing Graphs

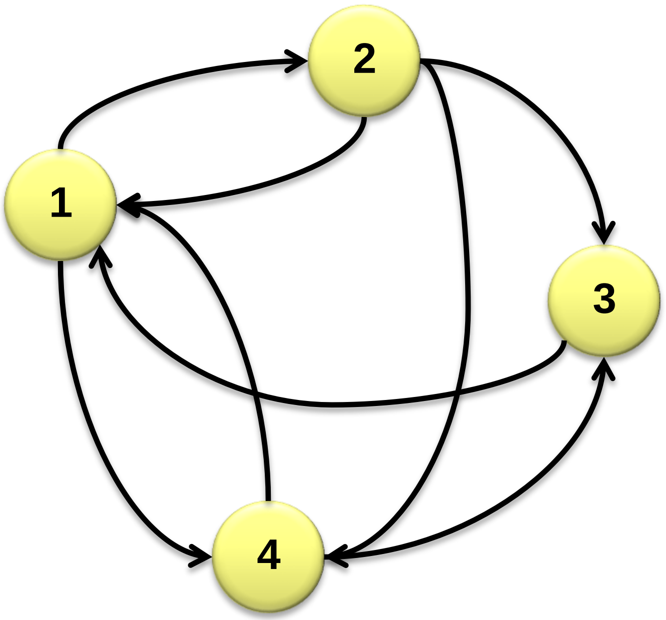
- $G = (V, E)$
- Two common representations
  - Adjacency matrix
  - Adjacency list

Adjacency Matrices

Represent a graph as an $n \times n$ square matrix M

- $n = |V|$
- $M_{ij} = 1$ means a link from node i to j

	1	**2**	**3**	**4**
1	0	1	0	1
2	1	0	1	1
3	1	0	0	0
4	1	0	1	0



Adjacency Matrices: Critique

- Advantages:
  - Amenable to mathematical manipulation
  - Iteration over rows and columns corresponds to computations on outlinks and inlinks
- Disadvantages:
  - Lots of zeros for sparse matrices
  - Lots of wasted space

Adjacency Lists

Take adjacency matrices… and throw away all the zeros

	1	**2**	**3**	**4**
1	0	1	0	1
2	1	0	1	1
3	1	0	0	0
4	1	0	1	0

1: 2, 4
2: 1, 3, 4
3: 1
4: 1, 3

Adjacency Lists: Critique

- Advantages:
  - Much more compact representation
  - Easy to compute over outlinks
- Disadvantages:
  - Much more difficult to compute over inlinks

JSON Encoding

- JSON: JavaScript Object Notation
- lightweight data interchange format

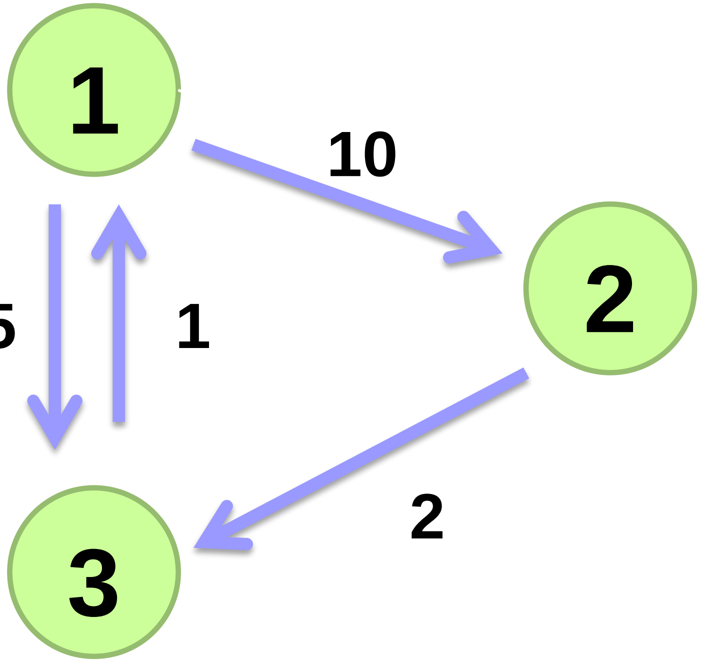
```
>>> import json
>>> json.dumps(['foo', {'bar': ('baz', None, 1.0, 2)}])
'["foo", {"bar": ["baz", null, 1.0, 2]}]'
```

```
>>> json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]')
[u'foo', {u'bar': [u'baz', None, 1.0, 2]}]
```

http://docs.python.org/library/json.html

Example Input

```
"1"     [0, ["2", 10], ["3", 5]]
"2"     [0, ["3", 2]]
"3"     [0, ["1", 1]]
```



JSON for MRJob

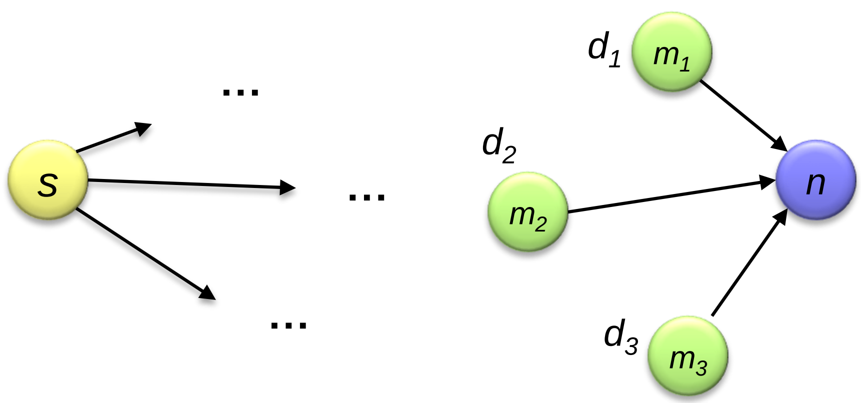
```python
from mrjob.job import MRJob
import sys
from mrjob.protocol import JSONProtocol

class jsonExample(MRJob):
    INPUT_PROTOCOL = JSONProtocol  # read the same format we write

    def mapper(self,key,value):
        yield key, value

    def reducer(self, key, values):
        for v in values:
            yield key, v

if __name__ == '__main__':
    jsonExample.run()
```

Single Source Shortest Path

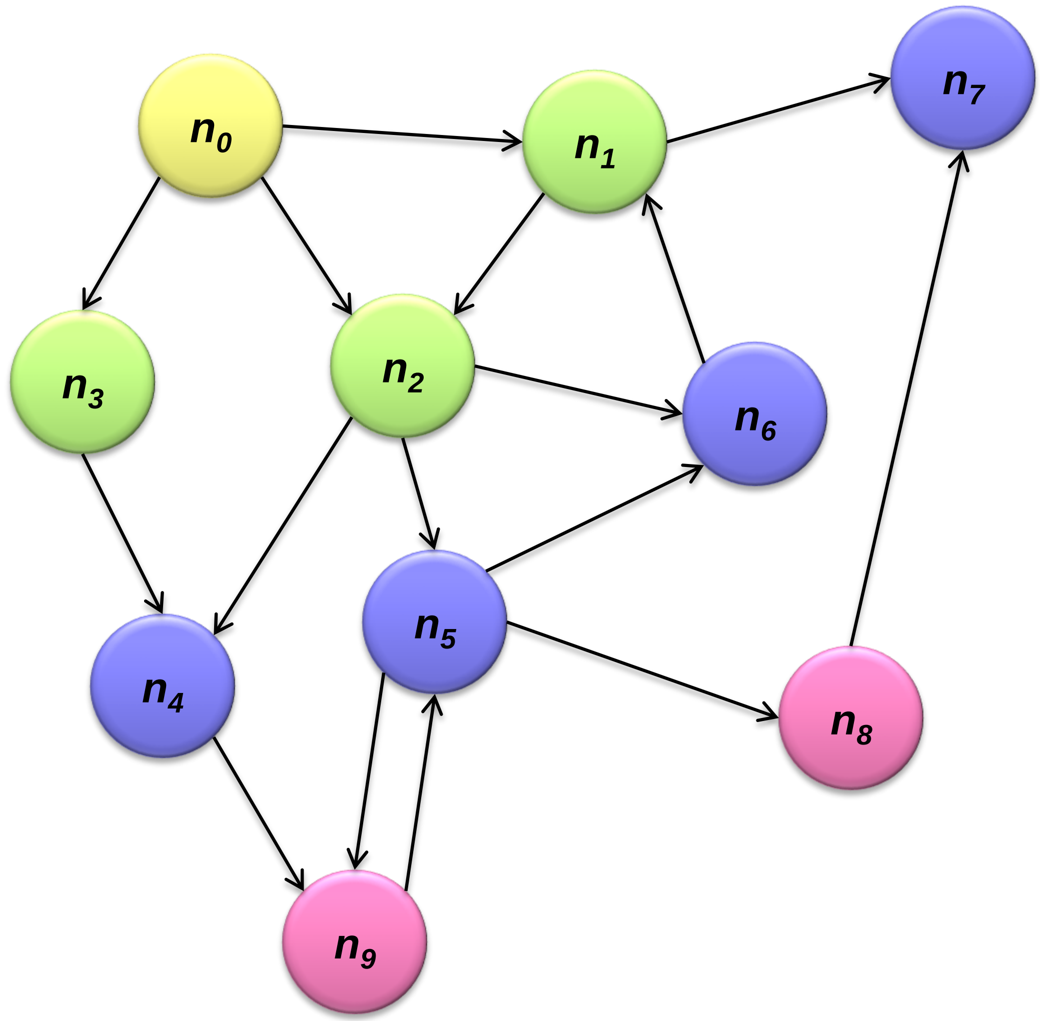
- **Problem:** find shortest path from a source node to one or more target nodes
  - Shortest might also mean lowest weight or cost
- MapReduce: parallel Breadth-First Search (BFS)

Finding the Shortest Path

- Consider simple case of equal edge weights
- Solution to the problem can be defined inductively
- Here's the intuition:
  - Define: *b* is reachable from *a* if *b* is on adjacency list of *a*
  - DistanceTo(s) = 0
  - For all nodes p reachable from s,
    DistanceTo(p) = 1
  - For all nodes n reachable from some other set of nodes M,
    DistanceTo(n) = 1 + min(DistanceTo(m), $m \in M$)



Visualizing Parallel BFS



From Intuition to Algorithm

- Data representation:
  - Key: node *n*
  - Value: *d* (distance from start), adjacency list (list of nodes reachable from *n*)
  - Initialization: for all nodes except for start node, $d = \infty$
- Mapper:
  - $\forall m \in$ adjacency list: emit (m, $d + 1$)
- Sort/Shuffle
  - Groups distances by reachable nodes
- Reducer:
  - Selects minimum distance path for each reachable node
  - Additional bookkeeping needed to keep track of actual path

Multiple Iterations Needed

- Each MapReduce iteration advances the “known frontier” by one hop
  - Subsequent iterations include more and more reachable nodes as frontier expands
  - Multiple iterations are needed to explore entire graph
- Preserving graph structure:
  - Problem: Where did the adjacency list go?
  - Solution: mapper emits (n, adjacency list) as well

BFS Pseudo-Code

class MAPPER
  method MAP(nid n, node N)
    $d \leftarrow N$.DISTANCE
    EMIT(nid n, N) ▷ Pass along graph structure
    for all nodeid $m \in N$.ADJACENCYLIST **do**
      EMIT(nid $m, d+1$) ▷ Emit distances to reachable nodes

class REDUCER
  method REDUCE(nid $m, [d_1, d_2, \ldots]$)
    $d_{min} \leftarrow \infty$
    $M \leftarrow \emptyset$
    for all $d \in$ counts $[d_1, d_2, \ldots]$ **do**
      if ISNODE(d) **then**
        $M \leftarrow d$ ▷ Recover graph structure
      else if $d < d_{min}$ **then** ▷ Look for shorter distance
        $d_{min} \leftarrow d$
    M.DISTANCE $\leftarrow d_{min}$ ▷ Update shortest distance
    EMIT(nid m, node M)

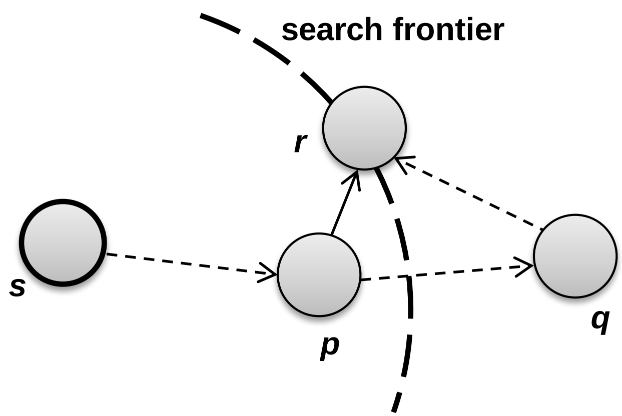
Weighted Edges

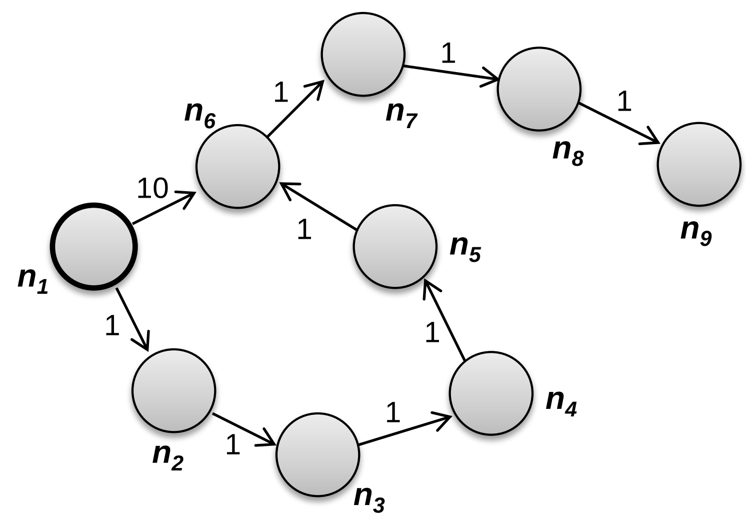
- Now add positive weights to the edges
- Simple change: adjacency list now includes a weight *w* for each edge
  - In mapper, emit (m, $d + w_p$) instead of (m, $d + 1$) for each node m

Stopping Criterion

- No updates in an iteration
- When a node is first “discovered”, we’ve **not** found the shortest path

Additional Complexities





Graphs and MapReduce

- Graph algorithms typically involve:
  - Performing computations at each node: based on node features, edge features, and local link structure
  - Propagating computations: “traversing” the graph
- Generic recipe:
  - Represent graphs as adjacency lists
  - Perform local computations in mapper
  - Pass along partial results via outlinks, keyed by destination node
  - Perform aggregation in reducer on inlinks to a node
  - Iterate until convergence: controlled by external “driver”
  - Don’t forget to pass the graph structure between iterations

Random Walks Over the Web

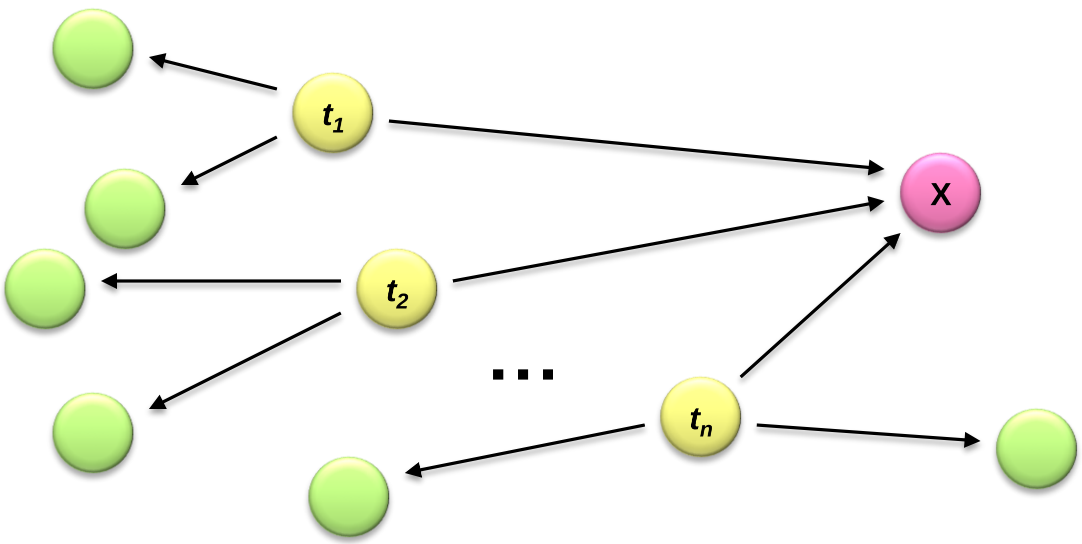
- Random surfer model:
  - User starts at a random Web page
  - User randomly clicks on links, surfing from page to page
- PageRank
  - Characterizes the amount of time spent on any given page
  - Mathematically, a probability distribution over pages
- PageRank captures notions of page importance
  - Correspondence to human intuition?
  - One of thousands of features used in web search

PageRank: Defined

Given page x with inlinks $t_1 \ldots t_n$, where

- $C(t)$ is the out-degree of t
- α is probability of random jump
- N is the total number of nodes in the graph

$$PR(x) = \alpha\left(\frac{1}{N}\right) + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$



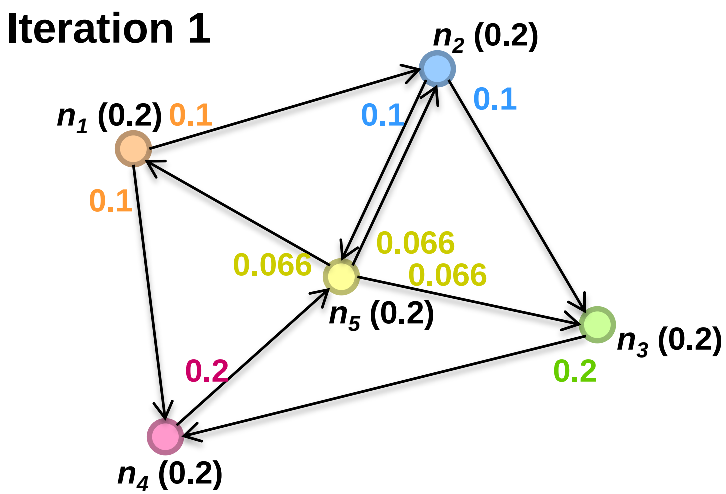
Computing PageRank

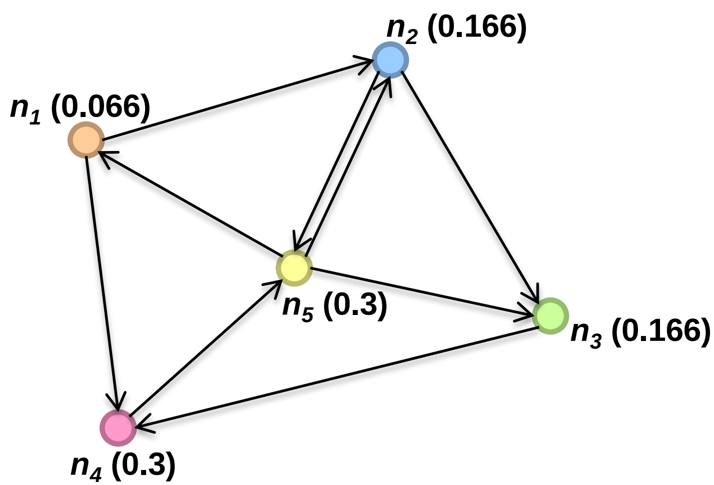
- Properties of PageRank
  - Can be computed iteratively
  - Effects at each iteration are local
- Sketch of algorithm:
  - Start with seed PR_i values
  - Each page distributes PR_i "credit" to all pages it links to
  - Each target page adds up "credit" from multiple in-bound links to compute PR_{i+1}
  - Iterate until values converge

Simplified PageRank

- First, tackle the simple case:
  - No random jump factor
  - No dangling links

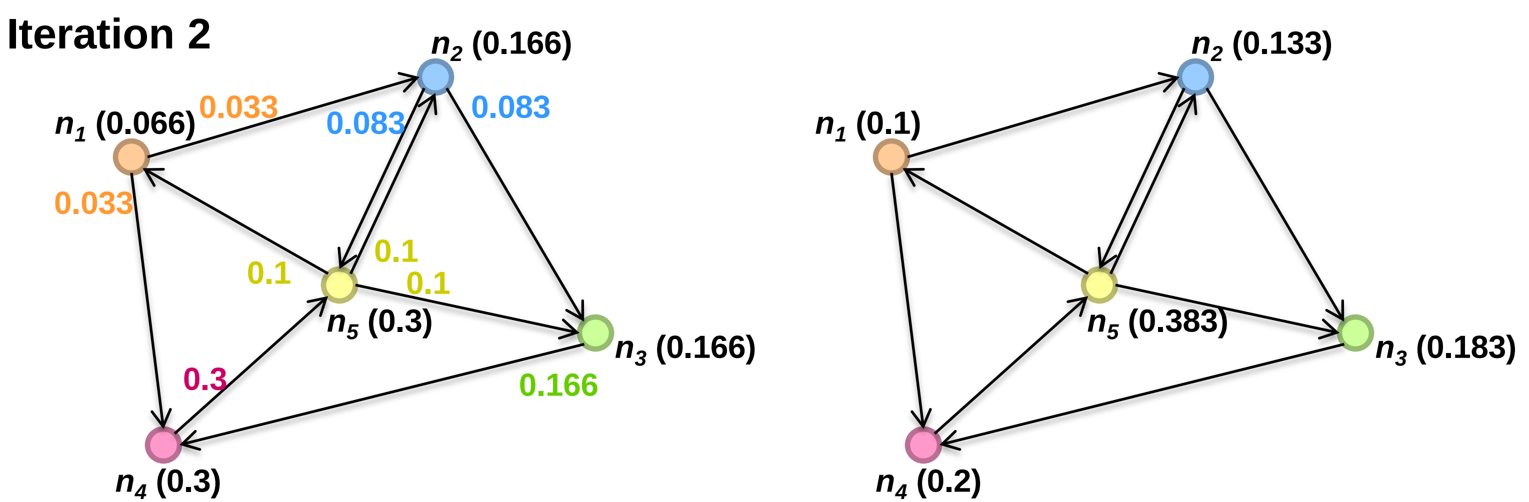
Sample PageRank Iteration (1)





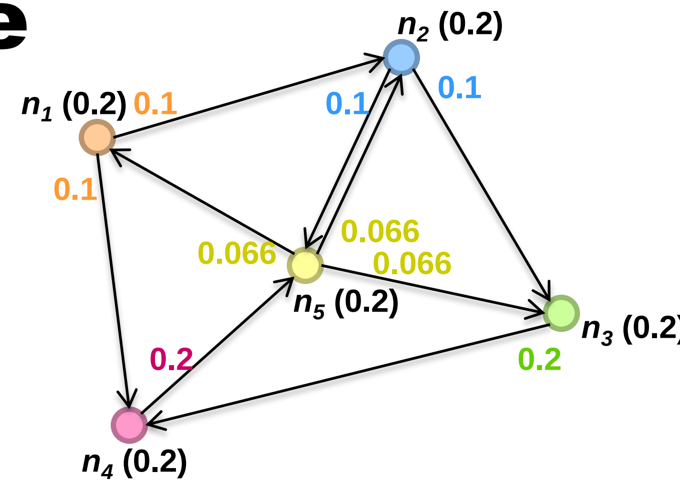
$$PR(x) = \alpha\left(\frac{1}{N}\right) + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$

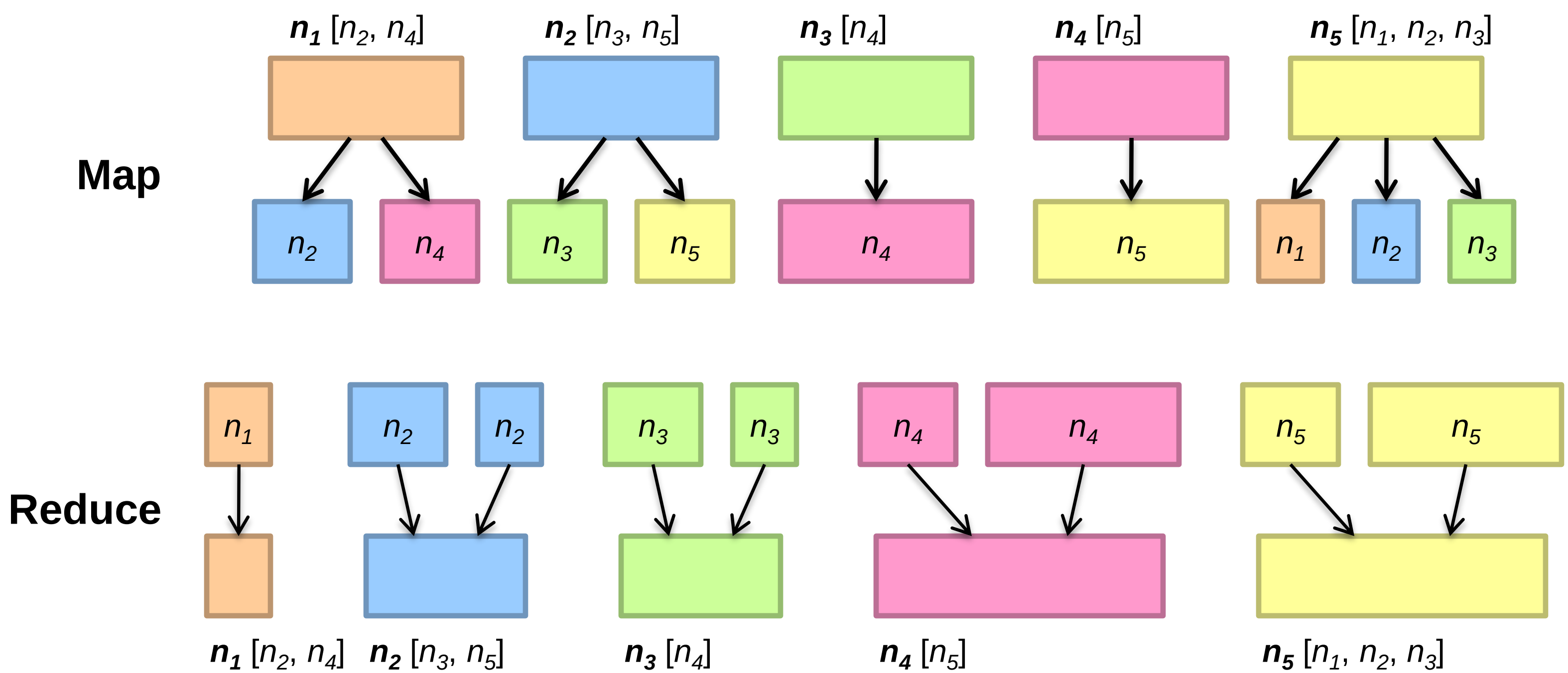
Sample PageRank Iteration (2)



$$PR(x) = \alpha\left(\frac{1}{N}\right) + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$

PageRank in MapReduce



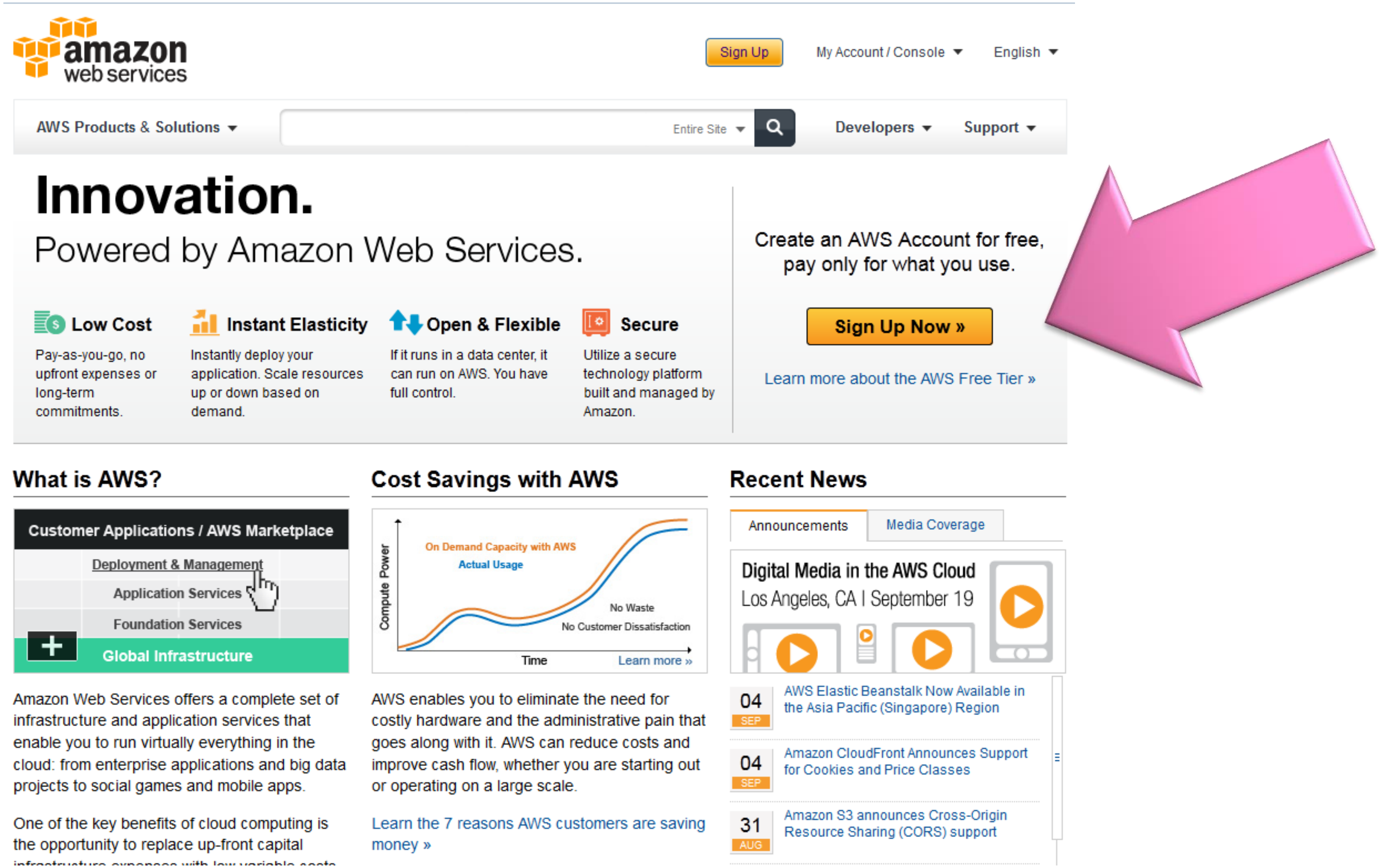


PageRank Pseudo-Code

```
class Mapper
    method Map(nid n, node N)
        p ← N.PageRank/|N.AdjacencyList|
        Emit(nid n, N)                          ▷ Pass along graph structure
        for all nodeid m ∈ N.AdjacencyList do
            Emit(nid m, p)                      ▷ Pass PageRank mass to neighbors

class Reducer
    method Reduce(nid m, [p1, p2, ...])
        M ← ∅
        for all p ∈ counts [p1, p2, ...] do
            if IsNode(p) then
                M ← p                           ▷ Recover graph structure
            else
                s ← s + p                       ▷ Sums incoming PageRank contributions
        M.PageRank ← s
        Emit(nid m, node M)
```

AMAZON ACCOUNT SETUP

- Go to aws.amazon.com



Register as a new user

amazon webservices™

Sign In or Create an AWS Account

You may sign in using your existing Amazon.com account or you can create a new a
"I am a new user."

My e-mail address is:

- (•) **I am a new user.**
- ( ) **I am a returning user and my password is:**

Sign in using our secure server

Forgot your password?

Has your e-mail address changed?

Learn more about AWS Identity and Access Management and AWS Multi-Factor Authentication, features that provide additional security for your AWS Account.

Fill in Account Form

- Name
- Address
- …

- Credit Card Information!
- You get a 100$ AWS credit code for your course work. If you use more, your credit card will be charged!

- Complete survey to register for credit code (see HW1)
- Wait for email reply with code

Redeem Credit

Account

- Account Activity
- AWS Identity and Access Management
- AWS Management Console
- Consolidated Billing
- DevPay
- Manage Your Account
- Payment Method
- Personal Information
- Security Credentials
- Usage Reports
- Billing Alerts
- Billing Preferences

AWS Credits

Welcome Verena Kaynig-Fittkau | Sign Out

If you have received a promotional credits code or a grant for using AWS, you can easily update your account here.

Credits Balance†: $100.00

Credits Details (as of last billing cycle -September 1, 2012)†

Credit Name	Applicable Products	Credits Used($)	Credits Remaining($)	Expiration Date
EDU_Cecka_HarvardCourse_Summer2012	Amazon Route 53, VPC, EC2, Elastic MapReduce, SES, CloudSearch, ElastiCache, AWS Elastic Beanstalk, AWS Data Transfer, SNS, Simple EDI, DynamoDB, CloudFront, RDS, Simple Notification Service, S3, and SimpleDB	0.00	100.00	08/31/2013

† Note: Credits are debited at the end of your monthly billing cycle. If your usage exceeds your credits balance, your payment method will be charged.

Enter your claim code below and click **Redeem**. We'll add the credits to your AWS account.

Redeem

Account Activity

Account Activity

Welcome Verena Kaynig-Fittkau | Sign Out

You are eligible for the AWS Free Usage Tier.See the Getting Started Guide AWS Free Usage Tier to learn how to get started with the free usage tier.

Monitor your estimated charges. Enable Now to begin setting billing alerts that automatically e-mail you when charges reach a threshold you define. Learn More

This Month's Activity as of September 7, 2012

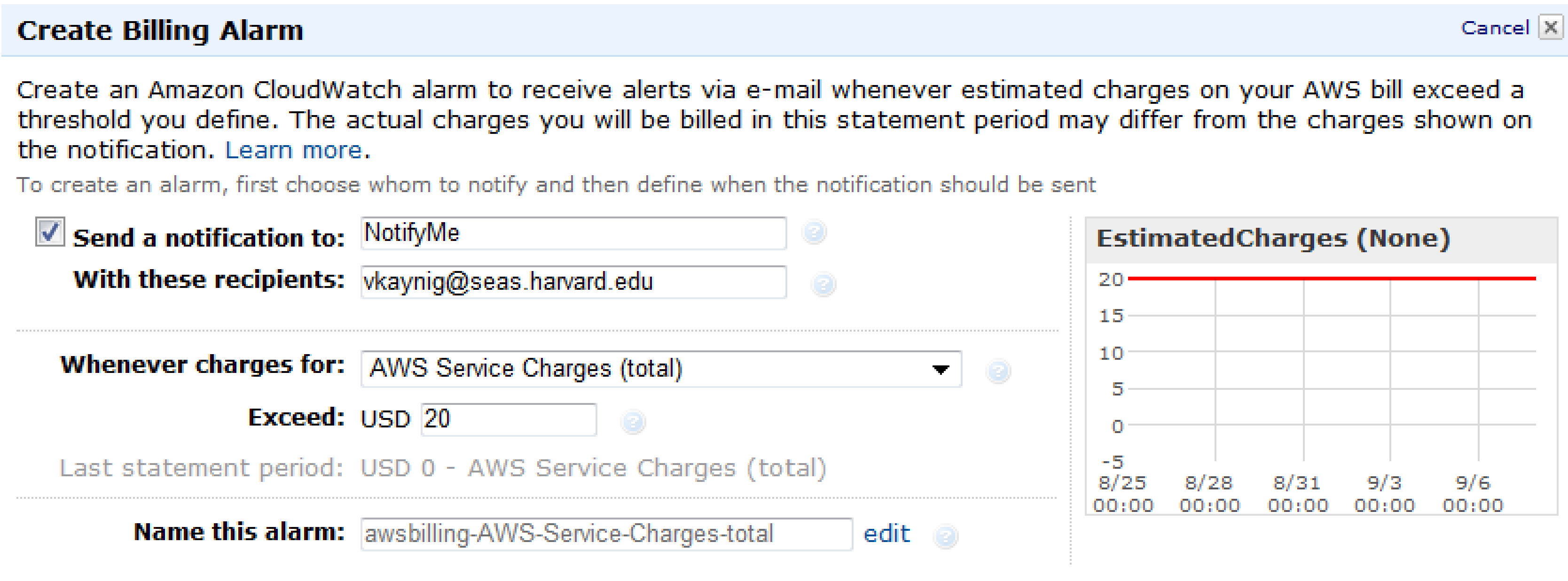
Credits

View Complete Credit Details»

Applicable Product(s)	Credits Remaining ($) §
Amazon Route 53, VPC, EC2, Elastic MapReduce, SES, CloudSearch, ElastiCache, AWS Elastic Beanstalk, AWS Data Transfer, SNS, Simple EDI, DynamoDB, CloudFront, RDS, Simple Notification Service, S3, and SimpleDB	$100.00

§ Remaining amounts shown are as of the end of the last statement period (September 1, 2012). Credits will be applied to your account at the close of the statement period.

Create Billing Alarm



- Up to 10 alarms per month free

How to Configure MRJob for EC2

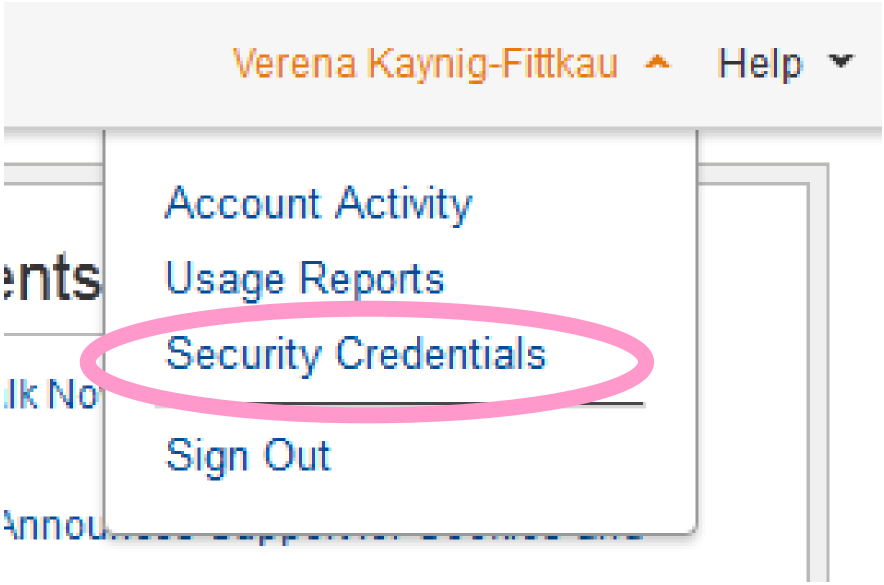
- Set environment variables:
  - AWS_ACCESS_KEY_ID
  - AWS_SECRET_ACCESS_KEY

- Use configuration file
  - Set environment variable to configuration file path
  - MRJOB_CONF

MRJOB CONFIG FILE

```
runners:
  emr:
    # be careful when editing this file
    # spaces vs tabs are important
    aws_access_key_id: MY_KEY_IS_SECURE
    # if you want to run in a different region
    # set it here
    # aws_region: us-west-1
    aws_secret_access_key: SO_IS_MY_PASSWORD
    # see the following link for different instance types.
    # use api names. http://aws.amazon.com/ec2/instance-types/
    ec2_instance_type: m1.small
    num_ec2_instances: 1
    check_emr_status_every: 5
```

How to Find your Keys



Access Credentials

Access Credentials

There are three types of access credentials used to authenticate your requests to AWS services: (a) access keys, (b) X.509 certificates, and (c) key pairs. Each access credential type is explained below.

Access Keys | X.509 Certificates | Key Pairs

Use access keys to make secure REST or Query protocol requests to any AWS service API. We create one for you when your account is created — see your access key below.

Your Access Keys

Created	Access Key ID	Secret Access Key	Status
September 6, 2012	[redacted]	Show	Active (Make Inactive)

Create a new Access Key

For your protection, you should never share your secret access keys with anyone. In addition, industry best practice recommends frequent key rotation.

Learn more about Access Keys

Running on Amazon

Before:

```
python myscript.py < inputfile.txt > outputfile.txt
```

After:

```
python myscript.py -r emr < inputfile.txt > outputfile.txt
```

Example Output

```
writing master bootstrap script to /tmp/P4.vkaynig.20120910.162239.47
Copying non-input files into s3://mrjob-8648d6ccf6b3dd79/tmp/P4.vkayn
.162239.472440/files/
Waiting 5.0s for S3 eventual consistency
Creating Elastic MapReduce job flow
Job flow created with ID: j-E7LU1A489C16
Job launched 5.1s ago, status STARTING
Job launched 10.2s ago, status STARTING
Job launched 15.3s ago, status STARTING
Job launched 20.4s ago, status STARTING: Starting instances
Job launched 25.4s ago, status STARTING: Starting instances
Job launched 30.5s ago, status STARTING: Starting instances
```

```
Job launched 185.0s ago, status STARTING: Starting instances
Job launched 190.1s ago, status STARTING: Starting instances
Job launched 195.2s ago, status STARTING: Configuring cluster software
Job launched 200.3s ago, status STARTING: Configuring cluster software
Job launched 205.4s ago, status STARTING: Configuring cluster software
Job launched 210.5s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 215.6s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 220.9s ago, status BOOTSTRAPPING: Running bootstrap actions
```

Example Output Part 2

```
Job launched 394.8s ago, status RUNNING: Running step (P4.vkaynig.20120910.1622
9.472440: Step 1 of 1)
Job completed.
Running time was 104.0s (not counting time spent waiting for the EC2 instances)
```

```
Counters from step 1:
  FileSystemCounters:
    FILE_BYTES_READ: 2195720
    FILE_BYTES_WRITTEN: 4424383
    S3_BYTES_READ: 1749525
    S3_BYTES_WRITTEN: 5128391
  Job Counters :
    Launched map tasks: 2
    Launched reduce tasks: 1
    Rack-local map tasks: 2
  Map-Reduce Framework:
    Combine input records: 0
    Combine output records: 0
    Map input bytes: 1743174
    Map input records: 172804
    Map output bytes: 4405039
    Map output records: 161841
    Reduce input groups: 156460
    Reduce input records: 161841
    Reduce output records: 156460
    Reduce shuffle bytes: 2228599
    Spilled Records: 323682
```

Good to Know

- Starting a job on Amazon can take a few minutes
- This is even the case for very small jobs
- Test locally!
- But, make sure code runs in cloud!

- Amazon takes some time to update your billing information.