Efficient Analysis of Big Data and Big Models through Distributed Computation

Benjamin E. Bagozzi & John Beieler

The Pennsylvania State University

Big Data Week Presentation Series
Penn State University
23 April 2013

Why Move to Distributed Computation?

"In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for more systems of computers."

-Grace Hopper

Hadoop

- An open source framework for distributed computing
- Two primary subprojects:
  - MapReduce: distributed data processing
  - HDFS: distributed data storage
- MapReduce jobs typically written in Java
- Hadoop Streaming: API for using MapReduce with other languages
  - E.g., Ruby, Python, R
- Additional subprojects: Pig, HBase, ZooKeeper, Hive, Chuckwa, etc.

MapReduce in Detail

- A two step paradigm for big data processing
- To implement:
  - Specify key-value pairs as input & output for each phase
  - Specify two functions: map function and reduce function
- Map phase: perform a transformation (e.g., field-extraction, parsing, filtering) on each individual piece of data (e.g., row of text, tweet, vector) and output a key-value pair
- Reduce phase: (1) sort and group output by key, (2) compute an aggregate function over the values associated with each key, (3) output aggregates to disk

Hadoop vs. Other Parallelization Approaches

- Other Parallelization Approaches:
  - Break tasks up by hand, submit pieces individually to HPCs
  - Split tasks via other parallelization paradigms (e.g., MPI)
- Hadoop Drawbacks:
  - More complex (debugging, configuration)
  - Less intuitive, steep learning curve
  - Availability & access
  - Bleeding edge
- Hadoop Benefits:
  - Flat scalability & efficient processing
  - Open Source
  - Integration with other languages, computing tasks
  - Reliable/robust big data storage and processing

Hadoop on SDSC's 'Gordon' Supercomputer

- Overviews of Gordon can be found here and here
- Available via the NSF's Extreme Science and Engineering Discovery Environment (XSEDE)
  - Register at XSEDE (free)
  - Request or join (via, e.g., PSU's Campus Champion Allocation) a Gordon-Allocation (not always free)
- Benefits:
  - Full base Hadoop framework available (see here)
  - Easy Hadoop job scheduling/submission via MyHadoop
- Drawbacks (as of April 2013):
  - Hadoop compliments (e.g., Hive, HBase, Pig) aren't available
  - Relevant libraries for (e.g.,) R and Python aren't installed

A Selection of Hadoop's Built-in (Java) Example Scripts

- **wordcount**: A map/reduce program that counts the words in the input files
- **aggregatewordcount**: Aggregate map/reduce program to count words in input files
- **multifilewc**: A job that counts words from several files
- **grep**: A map/reduce program that counts the matches of a regex in the input
- **dbcount**: An example job that counts the pageview counts from a database
- **randomwriter**: A map/reduce program that writes 10GB of random data per node
- **randomtextwriter**: A map/reduce program that writes 10GB random text per node
- **sort**: A map/reduce program that sorts the data written by the random writer
- **secondarysort**: An example defining a secondary sort to the reduce
- **teragen/terrasort/teravalidate**: terabyte generate/sort/transfer
- **pi**: A map/reduce program that estimates Pi using a monte-carlo method

For online tutorials on these, see here, here, and here.

Running Example: ICEWS News-Story Corpus

- 60 European and Middle Eastern countries
- All politically relevant stories, January 2001 to July 2011
- Document: Individual news-story (first 3-4 sentences)
- 6,681,537 Stories
- Removed: punctuation, stopwords, numbers, proper nouns, etc.
- Stemmed words

Applying wordcount to Entire News-Story Corpus

- What are the most frequent (stemmed) words?
- Map Stage: assign <key,value> pairs to corpus
  - Read in X-lines (stories) of text from corpus
  - Input <key,value>: <line-number, line-of-text>
  - Output <key,value>: <word, one>
- Reduce Stage: sum individual <key, value>'s from Map tasks
  - Input <key,value>: <word, one>
  - Output <key,value>: <word, occurrence>
- 1 Node→ 8 minutes; 4 Nodes→ 5 minutes
- 406,466 unique "words"

For online tutorials on Hadoop's wordcount, see here, here, and here.

Applying wordcount to Entire News-Story Corpus

- What are the most frequent (stemmed) words?

- Map Stage: assign <key,value> pairs to corpus
    - Read in X-lines (stories) of text from corpus
    - Input <key,value>: <line-number, line-of-text>
    - Output <key,value>: <word, one>

- Reduce Stage: sum individual <key, value>'s from Map tasks
    - Input <key,value>: <word, one>
    - Output <key,value>: <word, occurrence>

- 1 Node→ 8 minutes; 4 Nodes→ 5 minutes

- 406,466 unique "words"

For online tutorials on Hadoop's wordcount, see here, here, and here.

Applying wordcount to Entire News-Story Corpus

- What are the most frequent (stemmed) words?
- Map Stage: assign <key,value> pairs to corpus
  - Read in X-lines (stories) of text from corpus
  - Input <key,value>: <line-number, line-of-text>
  - Output <key,value>: <word, one>
- Reduce Stage: sum individual <key, value>'s from Map tasks
  - Input <key,value>: <word, one>
  - Output <key,value>: <word, occurrence>
- 1 Node$\rightarrow$ 8 minutes; 4 Nodes$\rightarrow$ 5 minutes
- 406,466 unique "words"

For online tutorials on Hadoop's wordcount, see here, here, and here.

Applying wordcount to Entire News-Story Corpus

- What are the most frequent (stemmed) words?
- Map Stage: assign <key,value> pairs to corpus
    - Read in X-lines (stories) of text from corpus
    - Input <key,value>: <line-number, line-of-text>
    - Output <key,value>: <word, one>
- Reduce Stage: sum individual <key, value>'s from Map tasks
    - Input <key,value>: <word, one>
    - Output <key,value>: <word, occurrence>
- 1 Node$\rightarrow$ 8 minutes; 4 Nodes$\rightarrow$ 5 minutes
- 406,466 unique "words"

For online tutorials on Hadoop's wordcount, see here, here, and here.

Applying wordcount to Entire News-Story Corpus

- What are the most frequent (stemmed) words?
- Map Stage: assign <key,value> pairs to corpus
    - Read in X-lines (stories) of text from corpus
    - Input <key,value>: <line-number, line-of-text>
    - Output <key,value>: <word, one>
- Reduce Stage: sum individual <key, value>'s from Map tasks
    - Input <key,value>: <word, one>
    - Output <key,value>: <word, occurrence>
- 1 Node→ 8 minutes; 4 Nodes→ 5 minutes
- 406,466 unique "words"

For online tutorials on Hadoop's wordcount, see here, here, and here.

Applying wordcount to Entire News-Story Corpus

- What are the most frequent (stemmed) words?
- Map Stage: assign <key,value> pairs to corpus
  - Read in X-lines (stories) of text from corpus
  - Input <key,value>: <line-number, line-of-text>
  - Output <key,value>: <word, one>
- Reduce Stage: sum individual <key, value>'s from Map tasks
  - Input <key,value>: <word, one>
  - Output <key,value>: <word, occurrence>
- 1 Node$\rightarrow$ 8 minutes; 4 Nodes$\rightarrow$ 5 minutes
- 406,466 unique "words"

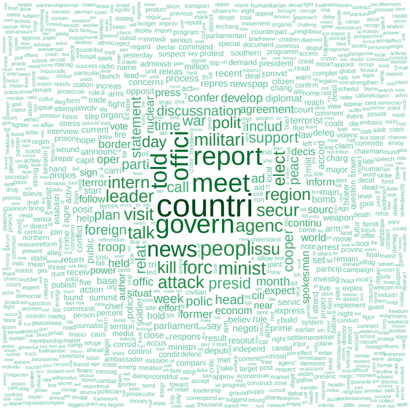
For online tutorials on Hadoop's wordcount, see here, here, and here.

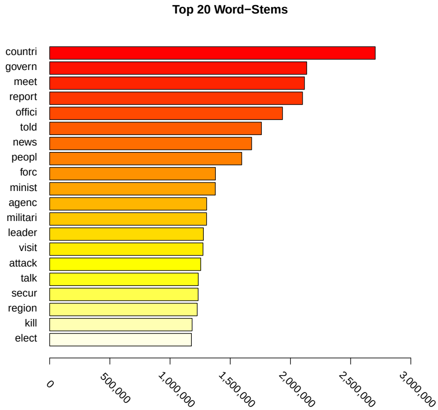
Applying wordcount to Entire News-Story Corpus

- What are the most frequent (stemmed) words?
- Map Stage: assign <key,value> pairs to corpus
    - Read in X-lines (stories) of text from corpus
    - Input <key,value>: <line-number, line-of-text>
    - Output <key,value>: <word, one>
- Reduce Stage: sum individual <key, value>'s from Map tasks
    - Input <key,value>: <word, one>
    - Output <key,value>: <word, occurrence>
- 1 Node→ 8 minutes; 4 Nodes→ 5 minutes
- 406,466 unique "words"

For online tutorials on Hadoop's wordcount, see here, here, and here.

Word-Stem Frequencies for ICEWS News Corpus





Cluster Analysis of Country News-Reports

- Do country-news reports cluster in interesting ways?
- K-Means on all 60 countries' news reports (1000 per country)
- Specify 60 clusters and set no. of iterations to 10
- Examine variation in cluster assignments across countries

Cluster Analysis of Country News-Reports

- Do country-news reports cluster in interesting ways?
- K-Means on all 60 countries' news reports (1000 per country)
- Specify 60 clusters and set no. of iterations to 10
- Examine variation in cluster assignments across countries

Cluster Analysis of Country News-Reports

- Do country-news reports cluster in interesting ways?
- K-Means on all 60 countries' news reports (1000 per country)
- Specify 60 clusters and set no. of iterations to 10
- Examine variation in cluster assignments across countries

Cluster Analysis of Country News-Reports

- Do country-news reports cluster in interesting ways?
- K-Means on all 60 countries' news reports (1000 per country)
- Specify 60 clusters and set no. of iterations to 10
- Examine variation in cluster assignments across countries

Cluster Analysis of Country News-Reports

- Do country-news reports cluster in interesting ways?
- K-Means on all 60 countries' news reports (1000 per country)
- Specify 60 clusters and set no. of iterations to 10
- Examine variation in cluster assignments across countries

K-Means with MapReduce/Java

- Map Stage: assign word values to (new) minimum distance clusters
    - Read in vectorized story-words (vv), and previous centers
    - For each word (v), apply distance function to find nearest center
    - Output <key,value>: <center$_i$,v>
- Reduce Stage: row bind and average <key, value>'s from Map tasks
    - Input <key,value>: <center$_i$, v >
    - Output: New center$_i$ $= mean(< center_i, vv >)$
- Non-Hadoop→1hr, 45 min; Hadoop→51 minutes

For running K-Means in Java, see here. For extending K-Means in Java to MapReduce and Hadoop, see here.

K-Means with MapReduce/Java

- Map Stage: assign word values to (new) minimum distance clusters
    - Read in vectorized story-words (vv), and previous centers
    - For each word (v), apply distance function to find nearest center
    - Output <key,value>: <center$_i$,v>
- Reduce Stage: row bind and average <key, value>'s from Map tasks
    - Input <key,value>: <center$_i$, v >
    - Output: New center$_i$ $= mean(< center_i, vv >)$
- Non-Hadoop→1hr, 45 min; Hadoop→51 minutes

For running K-Means in Java, see here. For extending K-Means in Java to MapReduce and Hadoop, see here.

K-Means with MapReduce/Java

- Map Stage: assign word values to (new) minimum distance clusters
  - Read in vectorized story-words (vv), and previous centers
  - For each word (v), apply distance function to find nearest center
  - Output <key,value>: <center$_i$,v>
- Reduce Stage: row bind and average <key, value>'s from Map tasks
  - Input <key,value>: <center$_i$, v >
  - Output: New center$_i$ $= mean(< center_i, vv >)$
- Non-Hadoop→1hr, 45 min; Hadoop→51 minutes

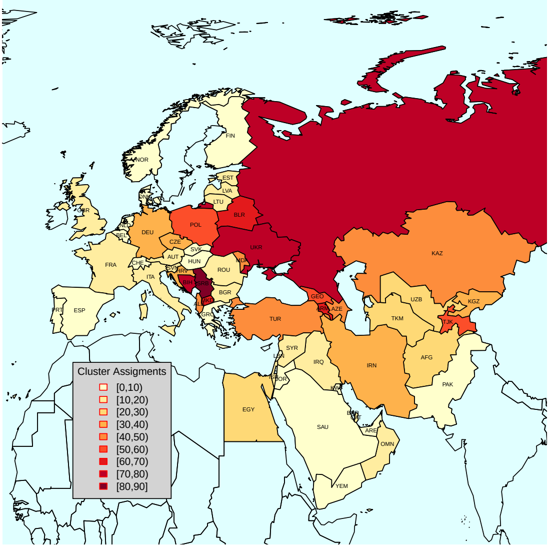
For running K-Means in Java, see here. For extending K-Means in Java to MapReduce and Hadoop, see here.

K-Means with MapReduce/Java

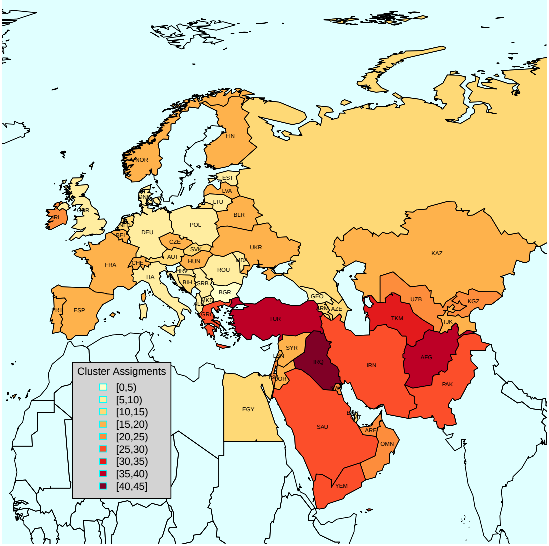
- Map Stage: assign word values to (new) minimum distance clusters
  - Read in vectorized story-words (vv), and previous centers
  - For each word (v), apply distance function to find nearest center
  - Output <key,value>: <center$_i$,v>
- Reduce Stage: row bind and average <key, value>'s from Map tasks
  - Input <key,value>: <center$_i$, v >
  - Output: New center$_i$ $= mean(< center_i, vv >)$
- Non-Hadoop→1hr, 45 min; Hadoop→51 minutes

For running K-Means in Java, see here. For extending K-Means in Java to MapReduce and Hadoop, see here.

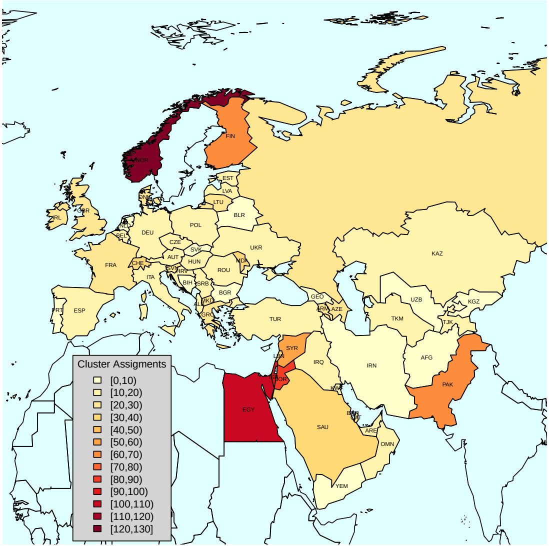
Cluster 4


Cluster Assigments
[0,10)
[10,20)
[20,30)
[30,40)
[40,50)
[50,60)
[60,70)
[70,80)
[80,90]
FIN
NOR
EST
LVA
LTU
BLR
POL
DEU
GBR
NLD
BEL
CZE
SVK
UKR
KAZ
AUT
HUN
FRA
CHE
ITA
ROU
BGR
ESP
PRT
TUR
GEO
AZE
UZB
KGZ
TKM
TJK
SYR
IRQ
IRN
AFG
PAK
EGY
SAU
ARE
OMN
YEM

Cluster 7


Cluster Assigments
[0,5)
[5,10)
[10,15)
[15,20)
[20,25)
[25,30)
[30,35)
[35,40)
[40,45]
FIN
NOR
EST
LVA
LTU
BLR
GBR
IRL
NLD
BEL
DEU
POL
CZE
SVK
UKR
AUT
HUN
MDA
FRA
CHE
ITA
HRV
BIH
SRB
ROU
BGR
MKD
ALB
GRC
TUR
GEO
ARM
AZE
KAZ
UZB
KGZ
TKM
TJK
PRT
ESP
SYR
LBN
IRQ
IRN
AFG
PAK
JOR
ISR
KWT
BHR
QAT
ARE
OMN
SAU
YEM
EGY

Cluster 40


Cluster Assigments
[0,10)
[10,20)
[20,30)
[30,40)
[40,50)
[50,60)
[60,70)
[70,80)
[80,90)
[90,100)
[100,110)
[110,120)
[120,130]
NOR
FIN
EST
LVA
LTU
BLR
POL
DEU
GBR
IRL
NLD
BEL
CZE
SVK
UKR
AUT
HUN
FRA
CHE
ROU
ITA
BIH
SRB
BGR
MKD
ALB
GRC
ESP
PRT
TUR
GEO
ARM
AZE
KAZ
UZB
TKM
KGZ
TJK
SYR
LBN
IRQ
IRN
AFG
PAK
JOR
KWT
EGY
SAU
QAT
ARE
OMN
YEM

Moving from Hadoop/Java to RHadoop

- Write and implement Hadoop jobs in R via Hadoop Streaming
- This requires three RHadoop packages: rhdfs, rmr, rhbase
    - Also requires additional prerequisite packages (e.g., rJava)
    - Also requires that you install and build Thrift
- Install & set-up Hadoop, RHadoop, and Streaming on EC2
- Use Amazon Elastic MapReduce (EMR) to run RHadoop via Streaming

Overviews of RHadoop, and installation info: here, here, here, and here.

Ready-Made Examples for RHadoop

- Basic data analysis
- Word count
- Logistic Regression
- K-Means (also here)
- Linear Least Squares

Getting Started on Hadoop

- For those interested in trying out Hadoop on Gordon...
- QuaSSIHadoop.zip
- Readme, .sh scripts, output files, error files, and all necessary input files for 4 basic Hadoop jobs:
  - Simple: a simple setup and usage example
  - TestDFS: depth-first search (DFS) benchmark
  - TeraSort: sorting benchmark
  - Wordcount: word frequencies

Questions?

Contact:

beb196@psu.edu

jub270@psu.edu