Distributed R for Big Data

Indrajit Roy, HP Labs
November 2013

Team:

Shivram

Erik

Kyungyong

Alvin

Rob

Vanish

© Copyright 2012 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.
A Big Data story

Once upon a time, a customer in distress had...

... 2+ billion rows of financial data (TBs of data)
... wanted to model defaults on mortgage and credit cards
... by running regression analysis

... Alas!
... traditional databases don’t support regression analysis
... custom code can take from hours to days

Moral of the story:
Customers need platform+programming model for complex analysis
Big data, complex algorithms

PageRank
(Dominant eigenvector)

Recommendations

Machine learning + Graph algorithms

Iterative Linear Algebra Operations
(40p K eigenvalues)

User Importance
(VerteX Centrality)
Example: PageRank using matrices

Simplified algorithm repeat \{ p = M*p \}

Linear Algebra Operations on Sparse Matrices

Power method
Dominant eigenvector

M = Web graph matrix
p = PageRank vector
Towards Distributed R

Variety (Efficiency)
- Machine learning, images, graphs
- SQL, database
- Search, sort

Volume (Scalability)
- R/Matlab
- RDBMS (col. store)
- Hadoop

Scale+ Complex Analytics

*very simplified view
Enter the world of R
R: Arrays for analysis

What is R?
R is a programming language and environment for statistical computing
• Array-oriented
• Millions of users, thousands of free packages

Traditional applications of R

Machine Learning  Graph Algorithms  Bioinformatics
R is used everywhere

...but R is

Not parallel
Not distributed
Limited by dataset size

Few parallel algorithms!
Enter Distributed R
Challenge 1: R has limited support for parallelism

- R is single-threaded
- Multi-process solutions offered by extensions

- Threads/processes share data through pipes or network
  - Time-inefficient (sending copies)
  - Space-inefficient (extra copies)
Challenge 2: R is memory bound

• Data needs to fit DRAM

• Current research solution:
  – Uses custom bigarray objects with limited functionality
  – Even simple operations like x+y may not work
Challenge 3: Sparse datasets cause load imbalance

Computation + communication imbalance!
Distributed R for Big Data

Challenges in scaling R

Programming model

Mechanisms

Applications and results
Large scale analytics frameworks

**Data-parallel frameworks – MapReduce/Dryad (2004)**
Process each *record* in parallel
Use case: Computing sufficient statistics, analytics queries

**Graph-centric frameworks – Pregel/GraphLab (2010)**
Process each *vertex* in parallel
Use case: Graphical models

**Array-based frameworks**
Process *blocks* of array in parallel
Use case: Linear Algebra Operations

*Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices.
Enhancement #1: Distributed data structures

- Relies on user defined partitioning
- Also support for distributed data-frames

darray
Enhancement #2: Distributed loop

- Express computations over partitions
- Execute across the cluster

foreach $f(x)$
Distributed PageRank

\[ M \leftarrow \text{darray}(\text{dim}=c(N,N), \text{blocks}=(s,N)) \]

\[ P \leftarrow \text{darray}(\text{dim}=c(N,1), \text{blocks}=(s,1)) \]

\[ \text{while(...)}{ \]
  \[ \text{foreach}(i,1:\text{len}, \]
    \[ \text{function}(p=\text{splits}(P,i), m=\text{splits}(M,i) \]
      \[ x=\text{splits}(P\_old), z=\text{splits}(Z,i) \} \]
      \[ p \leftarrow (m \times x) + z \]
      \[ \text{update}(p) \]
  \[ } \]

\[ P\_old \leftarrow P \]
Distributed PageRank

\[ M \leftarrow \text{darray}(\text{dim}=c(N,N), \text{blocks}=(s,N)) \]

\[ P \leftarrow \text{darray}(\text{dim}=c(N,1), \text{blocks}=(s,1)) \]

\[ \text{while}(...) \{ \]

\[ \text{foreach}(i,1:\text{len}, \]

\[ \text{function}(p=\text{splits}(P,i), m=\text{splits}(M,i)) \]

\[ x=\text{splits}(P_{\text{old}}), z=\text{splits}(Z,i)) \{ \]

\[ p \leftarrow (m*x)+z \]

\[ \text{update}(p) \]

\} \]

\[ P_{\text{old}} \leftarrow P \]

Execute function in a cluster

Pass array partitions
Distributed R for Big Data

Challenges in scaling R Programming model

Mechanisms

Applications and results
Architecture

• Scheduler: performs I/O and task scheduling
• Worker: executes tasks and I/O operations
Locality based computation: Part 1

\texttt{foreach}(i, 1\ldots 4, \text{ function}(p=\texttt{splits}(P, i)) \{\ldots\})

Ship functions to data

Task 1

\begin{tabular}{|c|c|}
\hline
P & 1 \\
\hline
\end{tabular}

M_1

Task 2

\begin{tabular}{|c|c|}
\hline
P & 2 \\
\hline
\end{tabular}

M_2

Task 3

\begin{tabular}{|c|c|}
\hline
P & 3 \\
\hline
\end{tabular}

M_3

Task 4

\begin{tabular}{|c|c|}
\hline
P & 4 \\
\hline
\end{tabular}

M_4
Locality based computation: Part 2

\[
\text{foreach}(i, 1:1, \text{ function}(p=\text{splits}(P)) \{ \ldots \})
\]
Distributed R for Big Data

Challenges in scaling R
Programming model
Mechanisms
Applications and results
# Applications in Distributed R

<table>
<thead>
<tr>
<th>Application</th>
<th>Algorithm</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageRank</td>
<td>Eigenvector calculation</td>
<td>41</td>
</tr>
<tr>
<td>Triangle counting</td>
<td>Top-K eigenvalues</td>
<td>121</td>
</tr>
<tr>
<td>Netflix recommendation</td>
<td>Matrix factorization</td>
<td>130</td>
</tr>
<tr>
<td>Centrality measure</td>
<td>Graph algorithm</td>
<td>132</td>
</tr>
<tr>
<td>SSSP</td>
<td>Graph algorithm</td>
<td>62</td>
</tr>
<tr>
<td>k-means</td>
<td>Clustering</td>
<td>71</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>Data mining</td>
<td>120</td>
</tr>
</tbody>
</table>

*LOC for core of the applications*
Distributed R has good performance

Algorithm: PageRank (power method)
Dataset: ClueWeb graph, 100M vertices, 1.2B edges, 20GB
Setup: 8 SL 390 servers, 8 cores/server, 96GB RAM

*Shorter is better*
Summary

Big data requires complex analysis: machine learning, graph processing, etc.

Matrices and arrays are surprisingly handy data structures.

Distributed R: simplifies distributed analytics.

Still, much remains: formalism, single node performance, compiler improvements, package contributions, …
Thank you

HPL Team:
Alvin AuYoung, Rob Schreiber, Vanish Talwar

Interns:
Shivaram Venkataraman (UC Berkeley), Erik Bodzsar (U. Chicago),
Kyungyong Lee (UFL)

HP Vertica Developers

Collaborators:
Prof. Andrew Chien (U. Chicago)

http://www.hpl.hp.com/research/distributedr.html