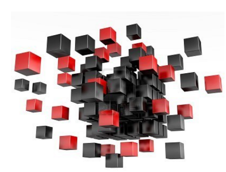
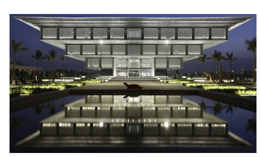
Presto/Blockus: Towards Scalable "R" Data Analysis

Andrew A. Chien
University of Chicago and Argonne National Laboratory
INRIA-UIUC-ANL Joint Institute
"Potential Collaboration"
November 19, 2012

LSSG Research Projects



- Blockus: Big Computations on small Memories



- GVR: Resilient Computation at massive scale

- 10x10: Systematic Heterogeneous, Energy-efficient Architecture

History: Towards Scalable R

- Presto (2010-)
  - Started by HP, collaboration with Berkeley
  - Shivaram Venkataraman, Indrajit Roy, Alvin AuYoung, Robert Schreiber, Erik Bodzsar
- Blockus (2011-)
  - Started by Uchicago, merged as a collaboration with Presto
  - Erik Bodzsar, Andrew Chien, Indrajit Roy, Robert Schreiber, Partha Ranganathan
- => One larger project Presto/Blockus

Outline

Motivation
Programming model
Applications and Results

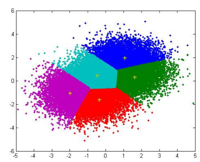
Big data analytics in R

What is R?

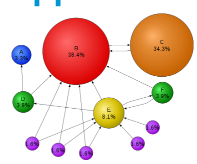
R is a programming language and environment for statistical computing

- Array-oriented
- Large, diverse user base

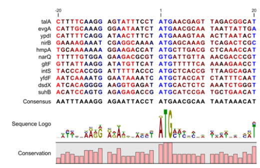
Examples of big data applications of R



Machine Learning



Graph Algorithms



Bioinformatics

5

Big Data, Complex Algorithms

PageRank
(Dominant eigenvector)

Machine learning + Graph algorithms

Iterative Linear Algebra Operations

Anomaly detection
(Top-K eigenvalues)

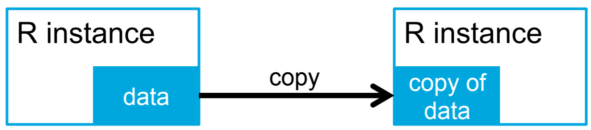
User Importance
(Vertex Centrality)

6 November 19, 2012

Challenge 1: R is not an efficient, parallel system

- 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)



7

Challenge 2: R is memory-bound

- Current research solution: bigmemory package
- Uses custom bigarray objects
- Relies on mmap and OS paging → Inefficient
- Need custom functions to access mmap contents

```
> x <- matrix(nrow = 2, ncol = 2, data = 1)
> x + 1
     [,1] [,2]
[1,]    2    2
[2,]    2    2
> y <- big.matrix(nrow = 2, ncol = 2, init = 1)
> y + 1
Error in y + 1 : non-numeric argument to binary operator
```

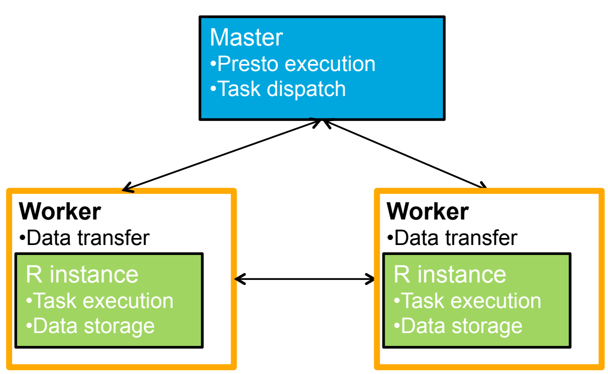
8

Outline

Motivation
Programming model
Applications and Results

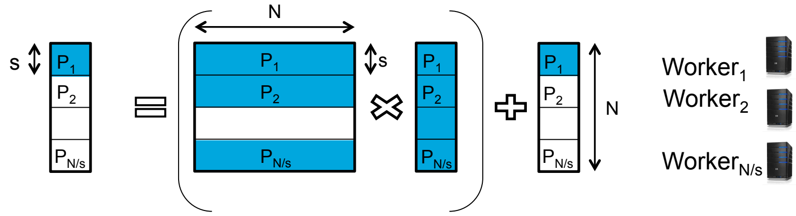
Presto: distributed execution engine for R

- Transparent data distribution and parallel execution
- Workers are still single-threaded and memory-bound



Added Feature #1: Distributed arrays

- Relies on user-defined data partitioning
- Computation is expressed over partitions



11

Added Feature #2: Versioning + Change Triggered Computation

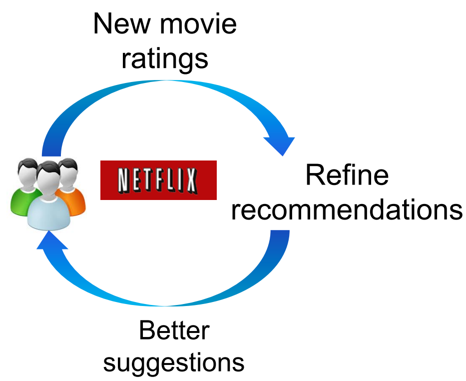
Mechanics of versioning

update: Increment version number

onchange: Bind a version number for the array before executing the handler

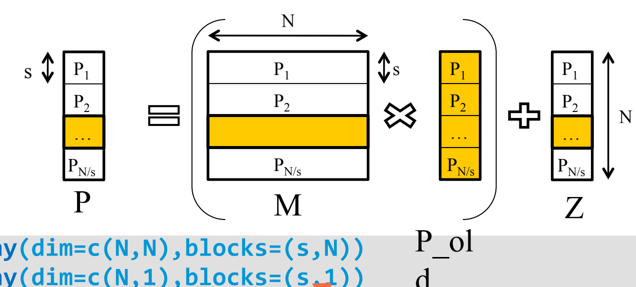
12 November 19, 2012

Incremental Updates



Incremental computation on consistent view of data

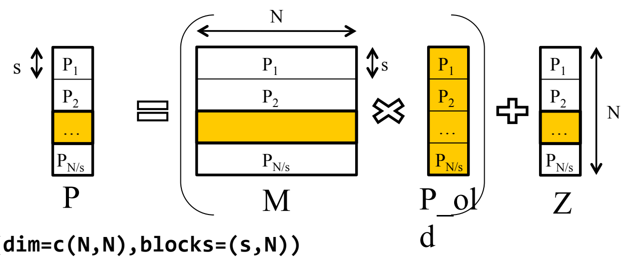
PageRank Using Presto




```
M ← darray(dim=c(N,N),blocks=(s,N))
P ← darray(dim=c(N,1),blocks=(s,1))
while(..){
  foreach(i,1:len,
    calculate(p=splits(P,i),m=splits(M,i),
              x=splits(P_old),z=splits(Z,i)) {
         p ← (m*x)+z
      }
  )
  P_old ← P
}
```

Create Distributed Array

PageRank Using Presto

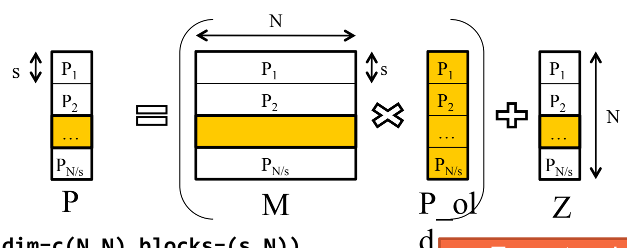



```
M ← darray(dim=c(N,N),blocks=(s,N))
P ← darray(dim=c(N,1),blocks=(s,1))
while(..){
  foreach(i,1:len,
     calculate(p=splits(P,i), m=splits(M,i),
               x=splits(P_old), z=splits(Z,i)) {
         p ← (m*x)+z
     }
  )
  P_old ← P
}
```

Execute function in a cluster

Pass array partitions

Incremental PageRank




```
M ← darray(dim=c(N,N),blocks=(s,N))
P ← darray(dim=c(N,1),blocks=(s1))
onchange(M) {
  while(..){
    foreach(i,1:len,
       calculate(p=splits(P,i), m=splits(M,i),
                 x=splits(P_old), z=splits(Z,i)) {
           p ← (m*x)+z
           update(p)
       })
    P_old ← P
}}
```

Execute when data changes

Update page rank vector

Outline

Motivation
Programming model
Applications and Results
- Scale Out
- Scale Vertical

Applications Implemented in Presto

Application	Algorithm	Presto LOC
PageRank	Eigenvector calculation	41
Triangle counting	Top-K eigenvalues	121
Centrality measure	Graph algorithm	132
k-path connectivity	Graph algorithm	30
k-means	Clustering	71
Sequence alignment	Smith-Waterman	64

Fewer than 140 lines of code

Achieving High Performance (w/ R runtime)

Good Parallelism

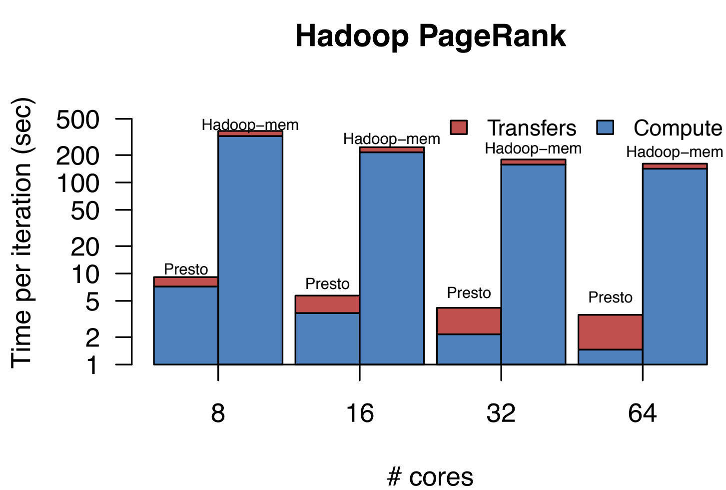
Load Balance

Efficient Exploitation of Multicore (SM, memory management)

... A whole list of other things....

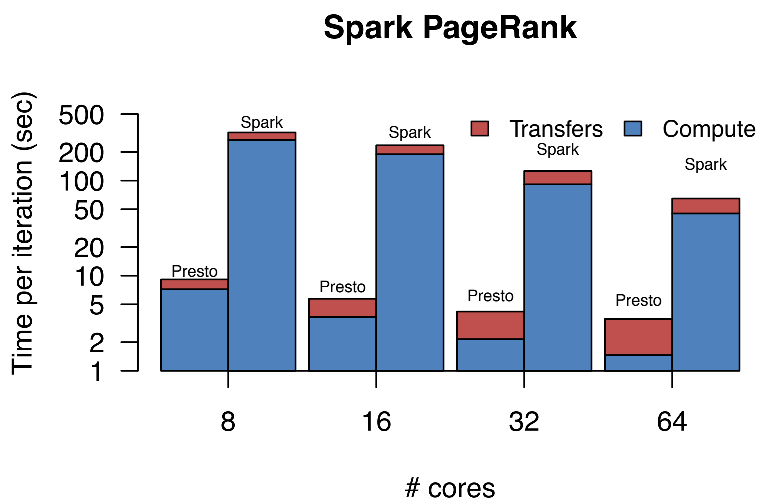
19

Presto vs Hadoop: PageRank



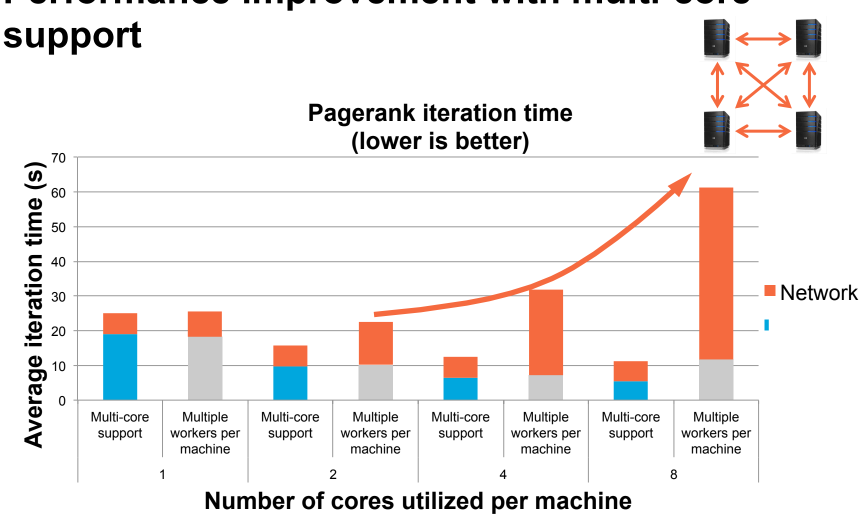
20

Presto vs. Spark PageRank



21

Performance improvement with multi-core support



Pagerank on 13 GB dataset (100M nodes, 1.2B edges) on 5 machines, varying the number of cores utilized per machine

Machine configuration: 12 cores, 96GB RAM

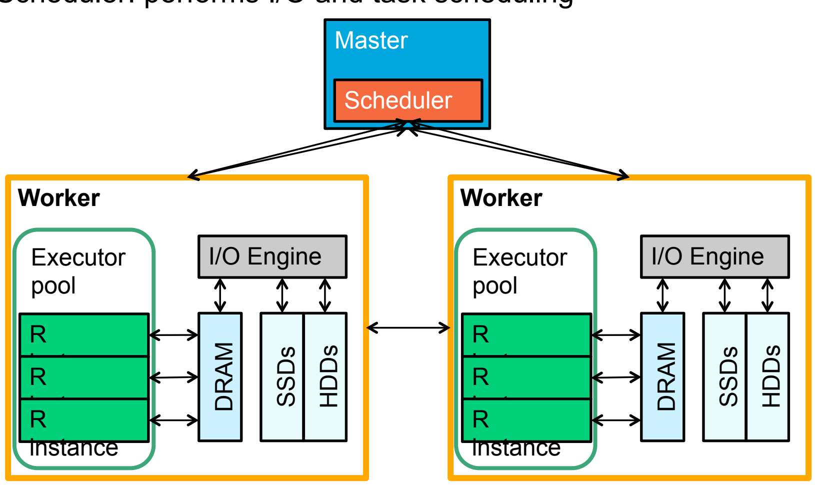
22

Outline

Motivation
Programming model
Applications and Results
- Scale Out
- **Scale Vertical**

Blockus architecture

- Worker I/O engine: executes all I/O operations
- Scheduler: performs I/O and task scheduling



Simple Blockus faster than OS paging

- Pagerank on 24.5GB dataset (134M nodes, 2B edges)
  - In-memory - 4 cores, 96 GB memory
  - Out-of-core
    - 8GB memory, 200MB/s SSD with 197 MB/s read
  - MMAP – OS based IO scheduling
  - Blockus – simple prefetch IO scheduling

	mmap	Blockus	In-memory
Average iteration time (seconds)	324	171	113
Speedup over mmap	1	1.89	2.86

25

Blockus Cost-effective

- Quantify out-of-core computation cost-effectiveness with EC2 prices
- Data set size is 24.5GB (in memory)
- Assuming SSD available (not true on EC2)

	Memory (total)	Cores (total)	Cost ($/hour)	Approx. iteration time (s)	Normalized cost
1 Large Instance	7.5GB	4	0.320	171	1
1 High-Memory Double XL Instance	34.2GB	12	0.900	70	1.14
4 Large Instances	30GB	16	1.280	49	1.14

← **Out-of-core**

26

Future Work: Scheduler policies

Presto scheduler
- Assumes everything fits in DRAM
- Schedules each task on worker which has most bytes of its input arrays
- Transfers non-local input data greedily (no network scheduling)

Blockus: better scheduling policies
- Load balancing (in memory)
- Intelligent Prefetching
- Intelligent computation prioritization
- Adaptive Caching

Summary

Broad focus on "Big Data" and applications
Surprising how broadaly useful linear algebra abstraction is
Easily express machine learning, graph algorithms

Challenges: Sparse matrices, Incremental data
Presto/Blockus –extends R
Exploit for Scale Out and Scale Vertical

Open source version soon

Related Work

Non-R Systems

- SciPy – parallel work (IPython1, MPI4py, parallel python, POSH),
- Star-P (MIT)
- MR/Hadoop, Bloom, Spark(Berkeley), Pig (Yahoo!), Ciel (Cambridge/)
- Pregel, Pregel 2? (Google)
- Graphlab, (CMU)

Parallel/Distributed R systems – threads, PVM/MPI, GridR

- Multicore: threads
- Rmpi, Snow: parallel abstraction over Rmpi, map/reduce
- Rmr: interface to Hadoop (by Revolution Analytics)
- GridR – Globus/Condor access, SwiftR
- Bigmemory: mmap arrays
- ...

