Challenges in distributed computing

• **Goal:** Need a simple API but with good performance

• **Solutions:**
  • Custom code: Very good performance, but suitable for only ninja programmers
  • R Packages with parallelism constructs: Data movement challenges
  • Interface to other systems (Hadoop, Spark): Need to learn external system’s API, installation, not native to R

• **Issues to tackle:**
  • Handle large datasets
  • Minimize data movement
Example: Cost of data movement

• Many applications reuse data
  • Multi-analysis on same data: load once, run many operations
  • Iterative algorithms: most machine learning + graph algorithms

• No persistent reference to distributed data -> High data movement costs

• Common workflow:
  • Load data at master->send to workers -> collect at master -> send to worker->repeat
Example: Cost of data movement

1. M
   - load(50GB)
   - send(50GB)
   - Worker
   - Worker
   - Worker

2. M
   - collect-results(1GB)
   - Worker
   - Worker

3. M
   - send(50GB)
   - Worker
   - Worker
   - Worker

Redundant data movement! No permanent data reference
Our approach

Open source package “distributedR”
Enhancement #1: Distributed data structures

- Extend existing R data-structures: darray, dframe, dlist
- Relies on user defined partitioning
Enhancement #2: Distributed loop

- Express computations over partitions
- Execute across the cluster

\[
\text{foreach} \quad f(x)
\]

Other option - dapply()?
Should allow functions on any pair of partitions?
Example: Writing distributed PageRank

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

while(..){
    foreach(i,1:len,
        function(p=\text{splits}(P,i),m=\text{splits}(M,i)
            x=\text{splits}(P\_old),z=\text{splits}(Z,i)) {
            p \leftarrow (m*x)+z
            \text{update}(p)
        }
    )
    P\_old \leftarrow P
}
Example: Writing distributed PageRank

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

while(\ldots){
    \text{foreach}(i,1:\text{len},
        \text{function}(p=\text{splits}(P,i), m=\text{splits}(M,i) \times x = \text{splits}(P\_old), z=\text{splits}(Z,i)) { }
    p \leftarrow (m\times x) + z \\
    \text{update}(p) \\
    } \\
    P\_old \leftarrow P \\
}
Behind the scenes
Distributed R architecture

- Programmers never think about physical location of data
- Master and scheduler: performs task and data scheduling
- Worker: executes tasks
Distributed R architecture

1. \texttt{Load\_data()}

Maintain data locality

Distributed R

Files or DB

P1

P2

P3

P4

© Copyright 2012 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.
Distributed R architecture

1. Load_data()

2. run-function(..)

Files or DB

Distributed R

Ship code to data
Distributed R architecture

1. Load_data()

2. run-function(..)

Files or DB

Distributed R

Ship data to code
Experience with applications
Examples of what we have written

• Embarrassingly parallel applications (add a field to each row)
• Simple statistics (mean, sum, colSums, ..)
• Regression (glm!), clustering algorithms
• Graph algorithms: PageRank
• Tree based algorithms: gradient boosting, randomforest ...
Scalability and performance are good

- Regression on billions of rows, analysis on 10B edge graphs, etc.
- Dataset size: 500+GB

**K-means clustering**

- Rows=60M to 480M, Cols=100
- Centers = 1,000
- Node = 12 core, 96GB RAM server
- Max. dataset size: 360 GB
Some observations
Summary: Keep it simple!

• Focus on data structures. Extend existing R data-structures.

• Add a parallelism construct that works with data-structures (chunk based apply(), foreach()?)

• May need to handle cases when chunk sizes are not known beforehand

• May need built in support for shuffle operation (e.g., groupBy)

• Extend scheduler to disks?
Thank you