Harp: Collective Communication on Hadoop

Judy Qiu, Indiana University
Outline

• Machine Learning on Big Data
• Big Data Tools
• Iterative MapReduce model
  • MDS Demo
• Harp
Machine Learning on Big Data

• Mahout on Hadoop
  • https://mahout.apache.org/

• MLlib on Spark
  • http://spark.apache.org/mllib/

• GraphLab Toolkits
  • http://graphlab.org/projects/toolkits.html
  • GraphLab Computer Vision Toolkit
Big Data Tools for HPC and Supercomputing

• MPI (Message Passing Interface, 1992)
  • Provide standardized function interfaces for communication between parallel processes.

• Collective communication operations
  • Broadcast, Scatter, Gather, Reduce, Allgather, Allreduce, Reduce-scatter.

• Popular implementations
  • MPICH (2001)
  • OpenMPI (2004)
    • [http://www.open-mpi.org/](http://www.open-mpi.org/)
MapReduce Model

Google MapReduce (2004)
- Jeffrey Dean et al. MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004.

Apache Hadoop (2005)

Apache Hadoop 2.0 (2012)
- Separation between resource management and computation model.
Key Features of MapReduce Model

• Designed for clouds
  • Large clusters of commodity machines

• Designed for big data
  • Support from local disks based distributed file system (GFS / HDFS)
  • Disk based intermediate data transfer in Shuffling

• MapReduce programming model
  • Computation pattern: Map tasks and Reduce tasks
  • Data abstraction: KeyValue pairs
## Applications & Different Interconnection Patterns

<table>
<thead>
<tr>
<th></th>
<th>(a) Map Only (Pleasingly Parallel)</th>
<th>(b) Classic MapReduce</th>
<th>(c) Iterative MapReduce</th>
<th>(d) Loosely Synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td><img src="image" alt="Input Diagram" /></td>
<td><img src="image" alt="Input Diagram" /></td>
<td><img src="image" alt="Input Diagram" /></td>
<td><img src="image" alt="Input Diagram" /></td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td><img src="image" alt="Output Diagram" /></td>
<td><img src="image" alt="Output Diagram" /></td>
<td><img src="image" alt="Output Diagram" /></td>
<td><img src="image" alt="Output Diagram" /></td>
</tr>
</tbody>
</table>

- **(a) Map Only (Pleasingly Parallel)**
  - CAP3 Gene Analysis
  - Smith-Waterman Distances
  - Document conversion (PDF -> HTML)
  - Brute force searches in cryptography
  - Parametric sweeps
  - PolarGrid MATLAB data analysis

- **(b) Classic MapReduce**
  - High Energy Physics (HEP) Histograms
  - Distributed search
  - Distributed sorting
  - Information retrieval
  - Calculation of Pairwise Distances for sequences (BLAST)

- **(c) Iterative MapReduce**
  - Expectation maximization algorithms
  - Linear Algebra
  - Data mining, includes K-means clustering
  - Deterministic Annealing Clustering
  - Multidimensional Scaling (MDS)
  - PageRank

- **(d) Loosely Synchronous**
  - Many MPI scientific applications utilizing wide variety of communication constructs, including local interactions
  - Solving Differential Equations and particle dynamics with short range forces

### Domain of MapReduce and Iterative Extensions

- No Communication
- Collective Communication
- MPI
What is Iterative MapReduce?

Iterative MapReduce

– Mapreduce is a Programming Model instantiating the paradigm of bringing computation to data

– Iterative Mapreduce extends Mapreduce programming model and support iterative algorithms for Data Mining and Data Analysis

– Is it possible to use the same computational tools on HPC and Cloud?

– Enabling scientists to focus on science not programming distributed systems
(Iterative) MapReduce in Context

Support Scientific Simulations (Data Mining and Data Analysis)


Security, Performance, Portal

Services, Workflow

High Level Language

Cross Platform Iterative MapReduce (Collectives, Fault Tolerance, Scheduling)

Distributed File Systems, Object Store, Data Parallel File System

Linux HPC Bare-system, Amazon Cloud Virtualization, Windows Server Bare-system, Azure Cloud Virtualization

CPU Nodes, GPU Nodes

Applications

Programming Model

Runtime Storage

Infrastructure

Hardware

Cross Platform Iterative MapReduce

Runtime Storage

Infrastructure

Hardware
Architecture (Upper)

Apache/Commercial Cloud (light) to HPC (darker)

Non Apache projects

Cross Cutting Capabilities

Monitoring: Ambari, Ganglia, Nagios, Inca (NA)

Security & Privacy

Distributed Coordination: Zooker, JGroups

Message Protocols: Thrift, Pachyderm (NA)

Data Analytics Libraries:

Machine Learning
Mahout, MLlib, MLbase
CompLearn (NA)

Statistics, Bioinformatics
R, Bioconductor (NA)

Imagery
ImageJ (NA)

Linear Algebra
Scalapack, PetSc (NA)

High Level (Integrated) Systems for Data Processing

Hive
(SQL on Hadoop)

Hcatalog Interfaces

Pig
(Procedural Language)

Shark
(SQL on Spark, NA)

MRQL
(SQL on Hadoop, Hama, Spark)

Impala (NA)
Cloudera (SQL on Hbase)

Swarm
(Log Google)

Parallel Horizontally Scalable Data Processing

Hadoop
(Map Reduce)

Spark
(Iterative MR)

Hama
(BSP)

Storm

S4

Samza
LinkedIn

Hive

Tez
(DAG)

Hama

Storm

S4

Samza
LinkedIn

Giraph

Parallel Batch
Stream
Graph

ABDS Inter-process Communication

Hadoop, Spark Communications
& Reductions

HPC Inter-process Communication

MPI (NA)

Harp Collectives (NA)

Hadoop, Spark Communications

Netty (NA)/ZeroMQ (NA)/ActiveMQ/Qpid/Kafka

Note: Harp and New Communication Layer
#### Cross Cutting Capabilities

<table>
<thead>
<tr>
<th>Extraction Tools</th>
<th>SQL</th>
<th>SciDB</th>
<th>NoSQL: Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIMA (Entities)</td>
<td>Tika (Content)</td>
<td>Phoenix (SQL on HBase)</td>
<td>HBase (Data on HDFS)</td>
</tr>
<tr>
<td>(Watson)</td>
<td></td>
<td></td>
<td>Accumulo (Data on HDFS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cassandra (DHT)</td>
</tr>
</tbody>
</table>

#### In memory distributed databases/caches:
- GORA (general object from NoSQL), Memcached (NA), Redis (key value), Hazelcast (NA), Ehcache (NA);

#### ORM Object Relational Mapping:
- Hiberate (NA), OpenJPA and JDBC Standard

#### NoSQL: Document
- MongoDB (NA)
- CouchDB
- Lucene

#### NoSQL: Key Value (all NA)
- Berkeley DB
- Azure Table
- Dynamo
- Moab
- Riak
- Dynamo

#### NoSQL: General Graph
- Neo4J
- Java Gnu (NA)
- Yarcdata Commercial (NA)
- Jena
- Sesame (NA)
- AllegroGraph Commercial
- RYA RDF on Accumulo

#### Data Transport
- BitTorrent, HTTP, FTP, SSH
- Globus Online (GridFTP)

#### ABDS Cluster Resource Management
- Mesos, Yarn, Helix, Llama(Cloudera)
- Condor, Moab, Slurm, Torque(NA) ……..

#### HPC Cluster Resource Management

#### ABDS File Systems
- User Level
- HPC File Systems (NA)

#### Interoperability Layer
- Whirr / JClouds
- Puppet/Chef/Boto/CloudMesh(NA)

#### DevOps/Cloud Deployment
- iROD

#### Security & Privacy
- Monitoring: Ambari, Ganglia, Nagios, Inca (NA)

#### Message Protocols: Thrift, Protobuf (NA)

#### Distributed Coordination
- ZooKeeper, JGroups

#### Monitoring:
- Ambari, Ganglia, Nagios, Inca (NA)

#### ORM Object Relational Mapping:
- Hibernate (NA), OpenJPA
- JDBC Standard

#### Extraction Tools
- ORM Object Relational Mapping: Hibernate (NA), OpenJPA
- JDBC Standard

#### IaaS System Manager
- Open Source
- Commercial Clouds
- OpenStack, OpenNebula, Eucalyptus, CloudStack, vCloud, Amazon, Azure, Google
Data Analysis Tools

*MapReduce optimized for iterative computations*

- **In-Memory**
  - Cacheable map/reduce tasks

- **Data Flow**
  - Iterative
  - Loop Invariant
  - Variable data

- **Thread**
  - Lightweight
  - Local aggregation

- **Map-Collective**
  - Communication patterns optimized for large intermediate data transfer

- **Portability**
  - HPC (Java)
  - Azure Cloud (C#)
  - Supercomputer (C++, Java)

*Twister: the speedy elephant*
Programming Model for Iterative MapReduce

- Distinction on loop invariant data and variable data (data flow vs. δ flow)
- Cacheable map/reduce tasks (in-memory)
- Combine operation

Main Program

while(..) {
    runMapReduce(..)
}

Loop Invariant Data
Loaded only once

Configure()

Map(Key, Value)

Reduce (Key, List<Value>)

Combine(Map<Key,Value>)

Cacheable map/reduce tasks (in memory)

Faster intermediate data transfer mechanism

Combiner operation to collect all reduce outputs
Twister Programming Model

Main program’s process space

- `configureMaps(...)`
- `configureReduce(...)`

while (`condition`){
- `runMapReduce(...)`

while
- `updateCondition()`
- `close()`

iterations

Cacheable map/reduce tasks

MapReduce

- `Map()`
- `Reduce()`
- `Combine()`

May scatter/broadcast <Key,Value> pairs directly

May merge data in shuffling

Communications/data transfers via the pub-sub broker network & direct TCP

- • Main program may contain many MapReduce invocations or iterative MapReduce invocations
Twister Architecture

Master Node
Twister Driver
Main Program

Twister Daemon

Worker Pool
map
reduce
Cacheable tasks

Worker Node
Local Disk

Pub/Sub Broker Network and Collective Communication Service

Scripts perform: Data distribution, data collection, and partition file creation

Worker Node
Local Disk
Well-known page rank algorithm [1]

- Used ClueWeb09 [2] (1TB in size) from CMU
- Hadoop loads the web graph in every iteration
- Twister keeps the graph in memory
- Pregel approach seems natural to graph-based problems

Data Intensive Kmeans Clustering

- **Image Classification**: 7 million images; 512 features per image; 1 million clusters; 10K Map tasks; 64G broadcasting data (1GB data transfer per Map task node); 20 TB intermediate data in shuffling.
Broadcast Comparison: Twister vs. MPI vs. Spark

Tested on IU Polar Grid with 1 Gbps Ethernet connection

At least a factor of 120 on 125 nodes, compared with the simple broadcast algorithm.

The new topology-aware chain broadcasting algorithm gives 20% better performance than best C/C++ MPI methods (four times faster than Java MPJ).

A factor of 5 improvement over non-optimized (for topology) pipeline-based method over 150 nodes.
Collective Model

- Harp (2013)
  - [https://github.com/jessezbj/harp-project](https://github.com/jessezbj/harp-project)
  - Hadoop Plugin (on Hadoop 1.2.1 and Hadoop 2.2.0)
  - Hierarchical data abstraction on arrays, key-values and graphs for easy programming expressiveness.
  - Collective communication model to support various communication operations on the data abstractions.
  - Caching with buffer management for memory allocation required from computation and communication
  - BSP style parallelism
  - Fault tolerance with check-pointing
Harp Design

Parallelism Model

• Architecture

MapReduce Model → Map-Collective Model

Shuffle

Collective Communication

Application

Framework

Resource Manager

MapReduce Applications

Map-Collective Applications

Harp

MapReduce V2

YARN
Hierarchical Data Abstraction and Collective Communication

- Array Table
- Edge Table
- Message Table
- Vertex Table
- Key Value Table

- Array Partition
- Edge Partition
- Message Partition
- Vertex Partition
- Key Value Partition

- Long Array
- Int Array
- Double Array
- Byte Array
- Vertices, Edges, Messages
- Key-Values
- Struct Object

- Commutable

Broadcast, Allgather, Allreduce, Regroup-(combine/reduce), Message-to-Vertex, Edge-to-Vertex
protected void mapCollective(KeyValReader reader, Context context)
throws IOException, InterruptedException {

ArrTable<DoubleArray, DoubleArrPlus> table =
    new ArrTable<DoubleArray, DoubleArrPlus>(0, DoubleArray.class, DoubleArrPlus.class);

if (this.isMaster()) {
    String cFile = conf.get(KMeansConstants.CFILE);
    Map<Integer, DoubleArray> cenDataMap = createCenDataMap(cParSize, rest, numCenPartition,
        vectorSize, this.getResourcePool());
    loadCentroids(cenDataMap, vectorSize, cFile, conf);
    addPartitionMapToTable(cenDataMap, table);

    rrTableBcast(table);
}
K-means Clustering Parallel Efficiency

WDA-MDS Performance on Big Red II

- WDA-MDS
  - Y. Ruan et. al, A Robust and Scalable Solution for Interpolative Multidimensional Scaling with Weighting. IEEE e-Science 2013.

- Big Red II
  - [http://kb.iu.edu/data/bcqf.html](http://kb.iu.edu/data/bcqf.html)

- Allgather
  - Bucket algorithm

- Allreduce
  - Bidirectional exchange algorithm
Parallel Efficiency

WDA-MDS Parallel Efficiency on Big Red II
Nodes: 8, 16, 32, 64, 128, with 32 Cores per Node
JVM settings: -Xmx42000M -Xms42000M -XX:NewRatio=1 -XX:SurvivorRatio=18
REEF

• Retainable Evaluator Execution Framework
• [http://www.reef-project.org/](http://www.reef-project.org/)
• Provides system authors with a centralized (pluggable) control flow
  • Embeds a user-defined system controller called the Job Driver
  • Event driven control
• Package a variety of data-processing libraries (e.g., high-bandwidth shuffle, relational operators, low-latency group communication, etc.) in a reusable form.
• To cover different models such as MapReduce, query, graph processing and stream data processing
Iterative Mapreduce - MDS Demo

I. Send message to start the job
II. Send intermediate results

- Input: 30k metagenomics data
- MDS reads pairwise distance matrix of all sequences
- Output: 3D coordinates visualized in PlotViz
Iterative Mapreduce - MDS Demo
KMeans Clustering Comparison
Hadoop vs. HDInsight vs. Twister4Azure

Shaded part is computation
Iterative MapReduce Models

• Twister (2010)
  • Jaliya Ekanayake et al. Twister: A Runtime for Iterative MapReduce. HPDC workshop 2010.
  • http://www.iterativemapreduce.org/
  • Simple collectives: broadcasting and aggregation.

• HaLoop (2010)
  • Yingyi Bu et al. HaLoop: Efficient Iterative Data Processing on Large clusters. VLDB 2010.
  • http://code.google.com/p/haloop/
  • Programming model $R_{i+1} = R_0 \cup (R_i \Join L)$
  • Loop-Aware Task Scheduling
  • Caching and indexing for Loop-Invariant Data on local disk
Model Composition

• Apache Spark (2010)
• Resilient Distributed Dataset (RDD)
• RDD operations
  • MapReduce-like parallel operations
• DAG of execution stages and pipelined transformations
• Simple collectives: broadcasting and aggregation
Graph Processing with BSP model

- Pregel (2010)
  - Grzegorz Malewicz et al. Pregel: A System for Large-Scale Graph Processing. SIGMOD 2010.

- Apache Hama (2010)
  - https://hama.apache.org/

- Apache Giraph (2012)
  - https://giraph.apache.org/
  - Scaling Apache Giraph to a trillion edges
We generalize the Map-Reduce concept to Map-Collective, noting that large collectives are a distinguishing feature of data intensive and data mining applications.

Collectives generalize Reduce to include all large scale linked communication-compute patterns.

MapReduce already includes a step in the collective direction with sort, shuffle, merge as well as basic reduction.
Future Work

- Run Algorithms on a much larger scale
- Provide Data Service on Clustering and MDS Algorithms
Acknowledgement

Bingjing Zhang Thilina Gunarathne     Fei Teng     Xiaoming Gao     Stephen Wu     Yuan Young
Prof. Haixu Tang
Bioinformatics

Prof. David Crandall
Computer Vision

Prof. Filippo Menczer
Complex Networks and Systems

SALSA HPC Group
http://salsahpc.indiana.edu
School of Informatics and Computing
Indiana University
Thank You!

Questions?