Iterative MapReduce Enabling HPC-Cloud Interoperability

IIT, Chicago, November 4, 2011

SALSA HPC Group
http://salsahpc.indiana.edu
Indiana University
Distributed Systems and Cloud Computing:
From Parallel Processing to the Internet of Things

Kai Hwang, Geoffrey Fox, Jack Dongarra
<table>
<thead>
<tr>
<th>Project Name</th>
<th>Team Members</th>
<th>Funding Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Twister</strong></td>
<td>Bingjing Zhang, Richard Teng</td>
<td>Funded by Microsoft, Indiana University's Faculty Research Support Program and NSF OCI-1032677 Grant</td>
</tr>
<tr>
<td><strong>Twister4Azure</strong></td>
<td>Thilina Gunarathne</td>
<td>Funded by Microsoft</td>
</tr>
<tr>
<td><strong>High-Performance Visualization Algorithms For Data-Intensive Analysis</strong></td>
<td>Seung-Hee Bae and Jong Youl Choi</td>
<td>Funded by NIH Grant 1RC2HG005806-01</td>
</tr>
<tr>
<td>Project Name</td>
<td>Authors</td>
<td>Funding Details</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DryadLINQ CTP Evaluation</td>
<td>Hui Li, Yang Ruan, and Yuduo Zhou</td>
<td>Funded by Microsoft</td>
</tr>
<tr>
<td>Cloud Storage, FutureGrid</td>
<td>Xiaoming Gao, Stephen Wu</td>
<td>Funded by Indiana University's Faculty Research Support Program and Natural Science Foundation Grant 0910812</td>
</tr>
<tr>
<td>Million Sequence Challenge</td>
<td>Saliya Ekanayake, Adam Hughes, Yang Ruan</td>
<td>Funded by NIH Grant 1RC2HG005806-01</td>
</tr>
<tr>
<td>Cyberinfrastructure for Remote Sensing of Ice Sheets</td>
<td>Jerome Mitchell</td>
<td>Funded by NSF Grant OCI-0636361</td>
</tr>
</tbody>
</table>
“In the last two decades advances in computing technology, from processing speed to network capacity and the Internet, have revolutionized the way scientists work.

From sequencing genomes to monitoring the Earth's climate, many recent scientific advances would not have been possible without a parallel increase in computing power - and with revolutionary technologies such as the quantum computer edging towards reality, what will the relationship between computing and science bring us over the next 15 years?”
Evolving Science

- Thousand years ago:
  science was **empirical**
  describing natural phenomena

- Last few hundred years:
  **theoretical** branch
  using models, generalizations

- Last few decades:
  a **computational** branch
  simulating complex phenomena

- Today:
  **data exploration (eScience)**
  synthesizing theory, experiment and computation with advanced data management and statistics
  → new algorithms!
Paradigm Shift in Data Intensive Computing
(Iterative) MapReduce in Context

Support Scientific Simulations (Data Mining and Data Analysis)


Applications

Programming Model

Runtime

Storage

Infrastructure

Hardware

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

Windows Server Bare-system

Azure Cloud

Grid Appliance

Virtualization

Runtime Storage Services and Workflow

Object Store

CPU Nodes

GPU Nodes
Iterative MapReduce Enabling HPC-Cloud Interoperability

Scientific Applications

Iterative MapReduce Runtime

Large Scale Infrastructure

HPC Cluster

Windows Azure

amazon web services

Twister
Cross Platform Iterative MapReduce

Twister for Azure Cloud

Main program may contain many MapReduce invocations or iterative MapReduce invocations

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

Worker Nodes

Cacheable map/reduce tasks

HPC Cluster

Windows Azure

amazon web services

INDIANA UNIVERSITY
BLOOMINGTON

SA-SA HPC
What are the challenges?

Providing both cost effectiveness for both computation and storage, and a programming model that is capable of handling the incredible increases in dataset sizes, in particular for data-intensive applications.

(large-scale data analysis for Data Intensive applications)

Research issues:

- portability between HPC and Cloud systems
- fault tolerance
- data locality
  - its impact on performance
  - the factors that affect data locality;
  - the maximum degree of data locality that can be achieved.
- factors beyond data locality to improve performance
  - task granularity and load balance
    - In MapReduce, task granularity is fixed.
    - This mechanism has two drawbacks
      1. limited degree of concurrency
      2. load unbalancing resulting from the variation of task execution time.
- scaling performance
- ...
Data Center vs Supercomputers

Scale
- Blue Waters = 40K 8-core “servers”
- Road Runner = 13K cell + 6K AMD servers
- MS Chicago Data Center = 50 containers = 100K 8-core servers.

Network Architecture
- Supercomputers: CLOS “Fat Tree” infiniband
  - Low latency – high bandwidth protocols
- Data Center: IP based
  - Optimized for Internet Access

Data Storage
- Supers: separate data farm
  - GPFS or other parallel file system
- DCs: use disk on node + memcache
New Software Architecture
Clouds hide Complexity

**Cyberinfrastructure**
Is “Research as a Service”

**SaaS**: Software as a Service
(e.g. Clustering is a service)

**PaaS**: Platform as a Service
IaaS plus core software capabilities on which you build SaaS
(e.g. Azure is a PaaS; MapReduce is a Platform)

**IaaS (HaaS)**: Infrastructure as a Service
(get computer time with a credit card and with a Web interface like EC2)
• Please sign and return your video waiver.
• Plan to arrive early to your session in order to copy your presentation to the conference PC.
• Poster drop-off is at Scholars Hall on Wednesday from 7:30 am – Noon. Please take your poster with you after the session on Wednesday evening.

Innovations in IP (esp. Open Source) Systems
Consistency models
Integration of Mainframe and Large Systems
Power-aware Profiling, Modeling, and…
IT Service and Relationship Management
Scalable Fault Resilience Techniques for Large…
New and Innovative Pedagogical Approaches
Data grid & Semantic web
Peer to peer computing
Autonomic Computing
Scalable Scheduling on Heterogeneous…
Hardware as a Service (HaaS)
Utility computing
Fault tolerance and reliability
Novel Programming Models for Large…
Auditing, monitoring and scheduling
Web services
Load balancing
Optimal deployment configuration
Software as a Service (SaaS)
High-performance computing
Virtualization technologies
Security and Risk
Cloud-based Services and Education
Middleware frameworks
Cloud /Grid architecture
Number of Submissions

0 10 20 30 40 50 60 70 80 90 100
Submissions

Topic

Number of Submissions

IEEE TCSC IEEE
Indiana University
Pervasive Technology Institute
<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache reference</td>
<td>0.5</td>
</tr>
<tr>
<td>Branch mispredict</td>
<td>5</td>
</tr>
<tr>
<td>L2 cache reference</td>
<td>7</td>
</tr>
<tr>
<td>Mutex lock/unlock</td>
<td>25</td>
</tr>
<tr>
<td>Main memory reference</td>
<td>100</td>
</tr>
<tr>
<td>Compress 1K w/cheap compression algorithm</td>
<td>3,000</td>
</tr>
<tr>
<td>Send 2K bytes over 1 Gbps network</td>
<td>20,000</td>
</tr>
<tr>
<td>Read 1 MB sequentially from memory</td>
<td>250,000</td>
</tr>
<tr>
<td>Round trip within same datacenter</td>
<td>500,000</td>
</tr>
<tr>
<td>Disk seek</td>
<td>10,000,000</td>
</tr>
<tr>
<td>Read 1 MB sequentially from disk</td>
<td>20,000,000</td>
</tr>
<tr>
<td>Send packet CA-&gt;Netherlands-&gt;CA</td>
<td>150,000,000</td>
</tr>
</tbody>
</table>
Programming Models and Tools

MapReduce in Heterogeneous Environment
Next Generation Sequencing Pipeline on Cloud

- Users submit their jobs to the pipeline and the results will be shown in a visualization tool.
- This chart illustrates a hybrid model with MapReduce and MPI. Twister will be an unified solution for the pipeline mode.
- The components are services and so is the whole pipeline.
- We could research on which stages of pipeline services are suitable for private or commercial Clouds.
Motivation

Data Deluge
Experiencing in many domains

Data Centered, QoS
Efficient and Proven techniques

Expand the Applicability of MapReduce to more classes of Applications

Map-Only
MapReduce
Iterative MapReduce
More Extensions

Map-Only
MapReduce
Iterative MapReduce

Input
map
Output

Input
map
reduce

Input
map
reduce

iterations

More Extensions

Pij
Twister v0.9

New Infrastructure for Iterative MapReduce Programming

- Distinction on static and variable data
- Configurable long running (cacheable) map/reduce tasks
- Pub/sub messaging based communication/data transfers
- Broker Network for facilitating communication
Main program’s process space

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

while(condition){

- runMapReduce(..)
  May send <Key,Value> pairs directly

- Combining operation

- updateCondition()

} //end while

close()

- Main program may contain many MapReduce invocations or iterative MapReduce invocations

Worker Nodes

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

Local Disk

Cacheable map/reduce tasks

Main program’s process space

Worker Nodes

Local Disk

Cacheable map/reduce tasks

Communications/data transfers via the pub-sub broker network & direct TCP
Worker Node
Local Disk
Worker Pool
Twister Daemon
Master Node
Twister Driver
Main Program
Pub/sub Broker Network

One broker serves several Twister daemons

Twister Daemon
map
reduce
Cacheable tasks

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

Worker Node
Local Disk
Daytona

Iterative MapReduce on Windows Azure

Microsoft has developed an iterative MapReduce runtime for Windows Azure, code-named "Daytona." Project Daytona is designed to support a wide class of data analytics and machine learning algorithms. It can scale out to hundreds of server cores for analysis of distributed data.

Project Daytona was developed as part of the eXtreme Computing Group’s Cloud Research Engagement Initiative, and made its debut at the Microsoft Research Faculty Summit. One of the most common requests we have received from the community of researchers in our program is for a data analysis and processing framework. Increasingly, researchers in a wide range of domains—such as healthcare, education, and environmental science—have large and growing data collections and they need simple tools to help them find signals in their data and uncover insights. We are making the Project Daytona MapReduce Runtime for Windows Azure download freely available, along with sample codes and instructional materials that researchers can use to set up their own large-scale,
Azure Queues for scheduling, Tables to store meta-data and monitoring data, Blobs for input/output/intermediate data storage.
- Programming model extensions to support broadcast data
- Merge Step
- In-Memory Caching of static data
- Cache aware hybrid scheduling using Queues, bulletin board (special table) and execution histories
- Hybrid intermediate data transfer
• Distributed, highly scalable and highly available cloud services as the building blocks.

• Utilize eventually-consistent, high-latency cloud services effectively to deliver performance comparable to traditional MapReduce runtimes.

• Decentralized architecture with global queue based dynamic task scheduling

• Minimal management and maintenance overhead

• Supports dynamically scaling up and down of the compute resources.

• MapReduce fault tolerance
Performance Comparisons

Cap3 Sequence Assembly

Smith Waterman Sequence Alignment

BLAST Sequence Search

Parallel Efficiency

Time to Process a Single Query File

Number of Query Files

Twister4Azure

Amazon EMR

Apache Hadoop

EC2-ClassicCloud

Cap3 Sequence Assembly

Time

Adjusted Time (s)

Num. of Cores * Num. of Files

Twister4Azure

Amazon EMR

Hadoop on Bare Metal

Apache Hadoop
Performance – Kmeans Clustering

Task Execution Time Histogram

Number of Executing Map Task Histogram

Strong Scaling with 128M Data Points

Weak Scaling
Performance – Multi Dimensional Scaling

- **Weak Scaling**
  - Execution Time vs. Number of Executing Map Tasks
  - **Azure Instance Type Study**
    - Amortised Cost vs. Execution Time
    - Number of Executing Map Task Histogram

- **Data Size Scaling**
  - Execution Time vs. Number of Data Points
  - Speedup gained using data cache
  - Scaling speedup
  - Increasing number of iterations

- Optional Step
  - Calculate BC
  - Calculate X
  - Calculate Stress

New Iteration

[Image: Diagram of the process flow]
PlotViz, Visualization System

- Parallel visualization algorithms (GTM, MDS, ...)
- Improved quality by using DA optimization
- Interpolation
- Twister Integration (Twister-MDS, Twister-LDA)

- Provide Virtual 3D space
- Cross-platform
- Visualization Toolkit (VTK)
- Qt framework
## GTM vs. MDS

<table>
<thead>
<tr>
<th>Purpose</th>
<th>GTM</th>
<th>MDS (SMACOF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Non-linear dimension reduction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Find an optimal configuration in a lower-dimension</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Iterative optimization method</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>Vector-based data</td>
<td>Non-vector (Pairwise similarity matrix)</td>
</tr>
<tr>
<td>Objective Function</td>
<td>Maximize Log-Likelihood</td>
<td>Minimize STRESS or SSTRESS</td>
</tr>
<tr>
<td>Complexity</td>
<td>$O(KN)$ ($K \ll N$)</td>
<td>$O(N^2)$</td>
</tr>
<tr>
<td>Optimization Method</td>
<td>EM</td>
<td>Iterative Majorization (EM-like)</td>
</tr>
</tbody>
</table>
Finding $K$ clusters for $N$ data points

- Relationship is a bipartite graph (bi-graph)
- Represented by $K$-by-$N$ matrix ($K << N$)

Decomposition for $P$-by-$Q$ compute grid

- Reduce memory requirement by $1/PQ$
**Parallel MDS**

- O(N^2) memory and computation required.
  - 100k data → 480GB memory
- Balanced decomposition of NxN matrices by P-by-Q grid.
  - Reduce memory and computing requirement by 1/PQ
- Communicate via MPI primitives

**MDS Interpolation**

- Finding approximate mapping position w.r.t. k-NN’s prior mapping.
- Per point it requires:
  - O(M) memory
  - O(k) computation
- Pleasingly parallel
- Mapping 2M in 1450 sec.
  - vs. 100k in 27000 sec.
  - 7500 times faster than estimation of the full MDS.
Interpolation extension to GTM/MDS

Full data processing by GTM or MDS is computing- and memory-intensive.

Two step procedure:

- **Training**: training by M samples out of N data
- **Interpolation**: remaining (N-M) out-of-samples are approximated without training
GTM/MDS Applications

PubChem data with CTD visualization by using MDS (left) and GTM (right)

About 930,000 chemical compounds are visualized as a point in 3D space, annotated by the related genes in Comparative Toxicogenomics Database (CTD).

Chemical compounds shown in literatures, visualized by MDS (left) and GTM (right)

Visualized 234,000 chemical compounds which may be related with a set of 5 genes of interest (ABCB1, CHRN2, DRD2, ESR1, and F2) based on the dataset collected from major journal literatures which is also stored in Chem2Bio2RDF system.
Twister-MDS Demo

This demo is for real time visualization of the process of multidimensional scaling (MDS) calculation.

We use Twister to do parallel calculation inside the cluster, and use PlotViz to show the intermediate results at the user client computer.

The process of computation and monitoring is automated by the program.
MDS projection of 100,000 protein sequences showing a few experimentally identified clusters in preliminary work with Seattle Children’s Research Institute
Twister-MDS Work Flow

I. Send message to start the job

II. Send intermediate results

III. Write data

IV. Read data

Local Disk

Client Node

PlotViz

MDS Monitor

ActiveMQ Broker

Twister-MDS

Master Node

Twister Driver

Twister
Twister-MDS Structure

Master Node
Twister Driver
Twister-MDS

Pub/Sub Broker Network

Twister Daemon
map
reduce

Twister Daemon
map
calculateBC
reduce

calculateStress

Worker Pool

Worker Node

MDS Output Monitoring Interface
Gene Sequences (N = 1 Million)

Select Reference

Reference Sequence Set (M = 100K)

Interpolative MDS with Pairwise Distance Calculation

Pairwise Alignment & Distance Calculation

Distance Matrix

Reference Coordinates

Multi-Dimensional Scaling (MDS)

Visualization

3D Plot
New Network of Brokers

A. Full Mesh Network

- Broker-Driver Connection
- Broker-Daemon Connection
- Broker-Broker Connection

B. Hierarchical Sending

- 7 Brokers and 32 Computing Nodes in total

C. Streaming

- 5 Brokers and 4 Computing Nodes in total
Performance Improvement

Twister-MDS Execution Time
100 iterations, 40 nodes, under different input data sizes
Broadcasting on 40 Nodes

(In Method C, centroids are split to 160 blocks, sent through 40 brokers in 4 rounds)
Twister New Architecture

Master Node
- Broker
- Configure Mapper
- Add to MemCache
- Map
- Reduce

Worker Node
- Broker
- Map broadcasting chain
- Cacheable tasks
- Reduce collection chain
- Twister Daemon

Twister Driver
- Twister Daemon

Worker Node
Chain/Ring Broadcasting

- **Driver sender:**
  - send broadcasting data
  - get acknowledgement
  - send next broadcasting data
  - ...

- **Daemon sender:**
  - receive data from the last daemon (or driver)
  - cache data to daemon
  - Send data to next daemon (waits for ACK)
  - send acknowledgement to the last daemon
Chain Broadcasting Protocol

Driver
- send
- get ack
- send
- get ack
- send
- get ack
- send
- get ack
- send
- get ack
- send
- get ack

Daemon 0
- receive
- handle data
- send
- ack
- receive
- handle data
- get ack
- send
- ack
- receive
- handle data
- get ack
- send
- ack
- receive
- handle data
- get ack
- send
- ack
- receive
- handle data
- get ack
- send
- ack

Daemon 1
- receive
- handle data
- send
- ack
- receive
- handle data
- get ack
- send
- ack
- receive
- handle data
- get ack
- send
- ack

Daemon 2
- receive
- handle data
- ack
- receive
- handle data
- ack

I know this is the end of Daemon Chain
I know this is the end of Cache Block
Broadcasting Time Comparison on 80 nodes, 600 MB data, 160 pieces

- Chain Broadcasting
- All-to-All Broadcasting, 40 brokers
# Applications & Different Interconnection Patterns

<table>
<thead>
<tr>
<th>Map Only</th>
<th>Classic MapReduce</th>
<th>Iterative Reductions</th>
<th>Loosely Synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Input map" /></td>
<td><img src="image2" alt="Input map" /></td>
<td><img src="image3" alt="Input map" /></td>
<td><img src="image4" alt="Input map" /></td>
</tr>
<tr>
<td><img src="image5" alt="Output" /></td>
<td><img src="image6" alt="Output" /></td>
<td><img src="image7" alt="Output" /></td>
<td><img src="image8" alt="Output" /></td>
</tr>
</tbody>
</table>

**Map Only**
- Input map
- Output

**Classic MapReduce**
- Input map
- Output
- reduce

**Iterative Reductions**
- Input map
- Output
- reduce
- iterations

**Loosely Synchronous**
- ![Pij](image9)

## CAP3 Analysis
- Document conversion (PDF -> HTML)
- Brute force searches in cryptography
- Parametric sweeps

## High Energy Physics (HEP)
- Histograms
- SWG gene alignment
- Distributed search
- Distributed sorting
- Information retrieval

## Expectation maximization algorithms
- Clustering
- Linear Algebra

## Many MPI scientific applications utilizing wide variety of communication constructs including local interactions
- Solving Differential Equations and particle dynamics with short range forces

### Examples:
- **CAP3 Gene Assembly**
- **PolarGrid Matlab data analysis**
- **Information Retrieval**
- **HEP Data Analysis**
- **Calculation of Pairwise Distances for ALU Sequences**
- **Kmeans**
- **Deterministic Annealing Clustering**
- **Multidimensional Scaling MDS**

---

**Domain of MapReduce and Iterative Extensions**

---

**MPI**

---

**SALSA**
Twister Futures

- Development of **library of Collectives** to use at Reduce phase
  - Broadcast and Gather needed by current applications
  - Discover other important ones
  - Implement efficiently on each platform – especially Azure
- Better **software message routing** with broker networks using asynchronous I/O with communication fault tolerance
- Support **nearby location of data and computing** using data parallel file systems
- Clearer application **fault tolerance** model based on implicit synchronizations points at iteration end points
- Later: Investigate **GPU** support
- Later: run time for **data parallel languages** like Sawzall, Pig Latin, LINQ
Convergence is Happening

Data intensive application (three basic activities): capture, curation, and analysis (visualization)

Cloud infrastructure and runtime

Parallel threading and processes
FutureGrid: a Grid Testbed

- IU Cray operational, IU IBM (iDataPlex) completed stability test May 6
- UCSD IBM operational, UF IBM stability test completes ~ May 12
- Network, NID and PU HTC system operational
- UC IBM stability test completes ~ May 27; TACC Dell awaiting delivery of components
• Switchable clusters on the same hardware (~5 minutes between different OS such as Linux+Xen to Windows+HPCS)
• Support for virtual clusters
• SW-G : Smith Waterman Gotoh Dissimilarity Computation as an pleasingly parallel problem suitable for MapReduce style applications
Demonstrate the concept of Science on Clouds using a FutureGrid cluster

- Top: 3 clusters are switching applications on fixed environment. Takes approximately 30 seconds.
- Bottom: Cluster is switching between environments: Linux; Linux + Xen; Windows + HPCS. Takes approximately 7 minutes
- SALSAHPC Demo at SC09. This demonstrates the concept of Science on Clouds using a FutureGrid iDataPlex.
Experimenting Lucene Index on HBase in an HPC Environment

- Background: data intensive computing requires storage solutions for huge amounts of data

- One proposed solution: HBase, Hadoop implementation of Google’s BigTable

<table>
<thead>
<tr>
<th>BasicInfo</th>
<th>ClassGrades</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Database</td>
</tr>
<tr>
<td>Office</td>
<td>Independent Study</td>
</tr>
<tr>
<td>t0 → aaa</td>
<td>t4 → A+</td>
</tr>
<tr>
<td>t1 → LH201</td>
<td>t5 → I</td>
</tr>
<tr>
<td>t2 → IE339</td>
<td>t6 → A</td>
</tr>
<tr>
<td>t3 → bbb</td>
<td>...</td>
</tr>
</tbody>
</table>

Column families: BasicInfo, ClassGrades
Qualifiers: Name, Office, Database, Independent Study
Row keys: aaa@indiana.edu, bbb@indiana.edu
Version timestamps: t0, t1, t2, t3, t4, t5, t6
System design

• Table schemas:
  - title index table: <term value> --> {frequencies:[<doc id>, <doc id>, ...]}
  - texts index table: <term value> --> {frequencies:[<doc id>, <doc id>, ...]}
  - texts term position vector table: <term value> --> {positions:[<doc id>, <doc id>, ...]}

• Natural integration with HBase
• Reliable and scalable index data storage
• Real-time document addition and deletion
• MapReduce programs for building index and analyzing index data
System implementation

- Experiments completed in the Alamo HPC cluster of FutureGrid
- MyHadoop -> MyHBase
- Workflow:
Index data analysis

Distribution of number of appearances in all documents

Distribution of record size in text index table
Education and Broader Impact

We devote a lot to guide students who are interested in computing
Education

We offer classes with emerging new topics

Together with tutorials on the most popular cloud computing tools
Broader Impact

Hosting workshops and spreading our technology across the nation

Giving students unforgettable research experience
Acknowledgement

SALSA HPC Group
Indiana University
http://salsahpc.indiana.edu