Big Data

By Michael Covert

April 2012
Agenda

• What is *Big Data* why are we discussing it?

• A brief history of *High Performance Computing*
  – Parallel processing
  – Algorithms
  – The No SQL camp

• The *Big Data* explosion
  – Science and technology
  – Social Networking and the Web

• Technology overview
  – The *Apache Hadoop* family
  – *Microsoft Dryad*
Agenda

• What is *Big Data* why are we discussing it?
• A brief history of High Performance Computing
  – Parallel processing
  – Algorithms
  – The No SQL camp
• The *Big Data* explosion
  – Science and technology
  – Social Networking and the Web
• Technology overview
  – The *Apache Hadoop* family
  – Microsoft Dryad
Agenda

• What is *Big Data* why are we discussing it?

• A brief history of High Performance Computing
  – Parallel processing
  – Algorithms
  – The No SQL camp

• The *Big Data* explosion
  – Science and technology
  – Social Networking and the Web

• Technology overview
  – The Apache Hadoop family

• Some considerations
  – Criticisms
  – Configuration Management
  – System Complexity
  – Performance and Tuning
  – Anticipated costs and benefits

• Questions and answers
What is *Big Data* why are we discussing it?

- Data, data, and more data
  - Facebook with 800+ million users generating something like 1.5 TB of data per day, storing over 70 PB of data – 21 PB in 2010, 30 PB in 2011, 50 PB in 2012?
  - Google indexes the Web

- Scientific data – megabytes, gigabytes, terabytes, petabytes, and now exabytes

- How can we ever process this amount of data?
What is **Big Data** why are we discussing it?

• How can we ever process this amount of data?
  – Have we finally collected more data than is feasible to use?

• Moore’s law of processing is overcome by the physics of data transfer limitations
  – At 100 MB/sec disk transfer speed, how long does it take to read 1 TB of data? How about 1 PB????
  – What if we can read from 10 disks simultaneously?
  – How about 100 disks? 1000? ...
What is *Big Data* why are we discussing it?

- Scaling up versus scaling out
  - Monolithic, proprietary systems are expensive to expand
  - But commoditized systems are hard to manage!
- Commoditization and cost
  - We have to consider disk and system failures as well so let’s include redundancy in our discussion.
  - The bigger they come, the harder they fall!
What is **Big Data** why are we discussing it?

- We are really discussing it because adoption is occurring at a staggering pace.
- Web scale corporate adoption began almost 8 years ago.
- Corporate adoption was slower, but has quickly become mainstream.
  - In October 2011, IT Management found 34% of those surveyed were planning on rolling out Hadoop within 12 months.
  - Review of hiring patterns shows rapid adoption is occurring.
- Discussions, debates, and downright rude and mean commentary is everywhere.
What is *Big Data* why are we discussing it?

- We are really discussing it because adoption is occurring at a staggering pace.
- Web scale corporate adoption began about 8 years ago.
- Corporate adoption was slower, but has quickly become mainstream.
  - In October 2011, IT Management found 34% of those surveyed were planning on rolling out Hadoop within 12 months.
  - Review of hiring patterns shows rapid adoption is occurring.
- Discussions, debates, and downright rude and mean commentary is everywhere.

The old guys are threatened. It’s got to be real!
What is **Big Data** why are we discussing it?

- Number of commercial vendors supporting **Hadoop** has increased.
- Number of cloud vendors offering PaaS has increased.

*Figure 1. Number of Client Searches for "Hadoop" on Gartner.com, March 2011 Through February 2012*

Source: Gartner (March 2012)
Big Data Processing – Some History

- Beowulf – early rather primitive attempt to use commodity computers to do parallel computing
- Sun Grid Engine, Apple Xgrid, IBM HPC, Oracle Grid Engine, and many, many more
- MPI, OpenMP(I), PVM and others
- Microsoft Dryad – Microsoft.NET based
  – Often described as a “superset of MapReduce.”
- And now, Hadoop
Big Data Processing – Some History

• So, what is Hadoop and what problems does it address?
  – Writing parallel programs is difficult and error prone when they share information and state
  – Monolithic systems are expensive to expand
  – Commoditized hardware is cheaper, but much more error-prone
    • Will the failure of a single component compromise the entire system?
Big Data Processing Systems

• **Hadoop – Java based Platform from Apache**
  – Distributed File System from Yahoo! Based on the Google Distributed Files System, and BigTable
  – MapReduce – Google’s famous parallelization architecture. A core component of Hadoop. Language independent.

• Now an emergent open source community with significant contributions from LinkedIn, Amazon.com, Twitter, Facebook, Google, and many, many others
  – We must mention **Cloudera** – the “Red Hat” of Hadoop
    • Cloudera Distros and the Cloudera Manager

• Hadoop is a family of products – HBase, Hive, Pig, Oozie, Zookeeper, Flume, Sqoop, Giraph, Mahout, and a plethora of other packages

• Just for fun, google Lucene and Nutch.
NoSQL

• We have all seen the specialized database explosion
  – Teradata machines
  – Multidimensional databases and cubes
    • OLAP, ROLAP, and HOLAP
    • Summarization and materialization
  – Column-oriented databases
  – Streaming databases

• Now we have NoSQL
NoSQL

• We have all seen the specialized database explosion
  – Teradata machines
  – Multidimensional databases and cubes
    • OLAP, ROLAP, and HOLAP
    • Summarization and materialization
  – Column-oriented databases
  – Streaming databases

• Now we have NoSQL
  – Holster your weapons. This really means “not only SQL” !!!!!
NoSQL

• ACID versus BASE
  – Atomic, consistent, isolated and durable
  – Basic availability, soft-state, and eventual consistency

• These are endpoints across a continuum!
NoSQL

Visual Guide to NoSQL Systems

Availability: Each client can always read and write.

Data Models
- Relational (comparison)
- Key-Value
- Column-Oriented/Tabular
- Document-Oriented

Pick Two

CA
- RDBMSs (MySQL, Postgres, etc)
- Aster Data
- Greenplum
- Vertica

AP
- Dynamo
- Voldemort
- Tokyo Cabinet
- KAI
- Cassandra
- SimpleDB
- CouchDB
- Riak

Consistency: All clients always have the same view of the data.

CP
- BigTable
- Hypertable
- Hbase
- MongoDB
- Terrastore
- Scalaris
- Berkeley DB
- MemcacheDB
- Redis

Partition Tolerance: The system works well despite physical network partitions.

http://blog.nahurst.com/visual-guide-to-nosql-systems
The Basic Hadoop Architecture

- One JobTracker per Hadoop instance
  - Clients “submit” jobs to the JobTracker
  - Validates job and adds to Job Queue
  - Schedules Job fragments (Map, Reduce, or intermediate steps) on nodes through a TaskTracker which runs on each node

- Hadoop Distributed File System (HDFS)
  - One NameNode per Hadoop instance
    - Manages all metadata
    - Manages DataNodes which store replicated versions of all data
  - All data is immutable!
Hadoop Architecture

Legend
- MapReduce Engine
- HDFS
- Data distribution

Java Libraries

Balancer

Disk Array

Rack #1

Rack #2

Rack #3
Hadoop Architecture

Legend
- MapReduce Engine
- HDFS

Data distribution

Java Libraries

Deliver the program to the data, not the data to the program!
Hadoop Architecture

Legend
MapReduce Engine
HDFS
Data distribution

Java Libraries

Deliver the program to the data, not the data to the program!
Use very large blocks of data

Task Tracker
Slot
Slot
Slot
Slot
Data Node

Rack #1
Disk Array
NameNode
JobTracker
Balancer

Task Tracker
Slot
Slot
Slot
Slot
Data Node

Rack #2
Disk Array

Task Tracker
Slot
Slot
Slot
Slot
Data Node

Rack #3
Disk Array

Task Tracker
Slot
Slot
Slot
Slot
Data Node

Data distribution

Java Libraries

Deliver the program to the data, not the data to the program!
Use very large blocks of data
Hadoop Architecture

Legend
- MapReduce Engine
- HDFS

Java Libraries

Balancer

Disk Array

Rack #1

Rack #2

Rack #3

Task Tracker
Data Node
Slot

JobTracker
NameNode

Data Node
Task Tracker
Slot

Job Queue

Data Node
Task Tracker
Slot

#1 - Create/submit Job
Hadoop Scheduling

#4 – Dispatch the task

First choice
Second choice
Third choice
#4 – Dispatch the task

First choice

Second choice

Third choice
#4 – Dispatch the task

First choice
Second choice
Third choice
Hadoop Scheduling

First choice
Second choice
Third choice

So, is there a problem? Rack #1

Rack #2

Rack #3
Hadoop Scheduling

- First choice
- Second choice
- Third choice

So, is there a problem?

- Rack #1
  - Algorithm design is important
  - Many tuning parameters

- Rack #2
  - Insensitive to resource capacity
  - Insensitive to current system status

- Rack #3
  - Mixed workloads cause problems
  - Single points of failure

• Algorithm design is important
• Many tuning parameters
• Insensitive to resource capacity
• Insensitive to current system status
• Mixed workloads cause problems
• Single points of failure

Map ready to run

JobTracker  Job Queue  NameNode

Disk Array
Hadoop Architecture

HDFS can redistribute Data

Legend

MapReduce Engine
HDFS

Task Tracker
Slot
Slot
Slot
Slot
Data Node

Balancer

Disk Array

Rack #1

Rack #2

Rack #3

Map task ready to run

JobTracker
Queue
NameNode

JobTracker

NameNode

Data Node

Task Tracker
Slot
Slot
Slot
Slot
Data Node

Disk Array
MapReduce Decomposition

User Job

Input reader

Map

Partition

Combine

Hadoop

Reduce

Output Writer

User Job

Map

Partition

Combine

Map

Partition

Map

Partition
MapReduce Decomposition

User Job

Input reader

Map

Map

Map

Partition

Partition

Partition

Combine

Combine

Reduce

Reduce

Output Writer

Hadoop
MapReduce Decomposition

Node 1
- Files loaded from local HDFS stores
- InputFormat
  - Split
  - Split
  - Split
  - Map
  - Map
  - Map
  - Partitioner
  - (sort)
  - Reduce
- Intermediate (k, v) pairs exchanged by all nodes
- Final (k, v) pairs
- OutputFormat
- Writeback to local HDFS store

Node 2
- Files loaded from local HDFS stores
- InputFormat
  - Split
  - Split
  - Split
  - Map
  - Map
  - Map
  - Partitioner
  - (sort)
  - Reduce
- Final (k, v) pairs
- OutputFormat
- Writeback to local HDFS store
The Hadoop Family

- HBase – a NoSQL database built on top of HDFS (Like Cassandra, Mongo, et al)
- Hive – a “SQL” queriable database of sorts that produces M/R “jobs”
- Pig – a scripting language that produces M/R “jobs”
- Flume and Sqoop for data acquisition
- Zookeeper for distributed systems management
- Oozie for workflows
- Mahout for machine learning
- GiRaph for graph processing
The Hadoop Family - HBase
Note that HBase overcomes the immutability restriction of HDFS
The Pig shell is named Grunt. It isn’t good enough to be called Oink!
The Hadoop Family – Pig and Hive

**Pig Latin**

countryys = load '/user/gharriso/PIG COUNTRIES' AS (country_id, country_name, country_subregion, region);
customers = load '/user/gharriso/PIG CUSTOMERS' AS (cust_id, first_name, last_name, gender, job, marital, postcode, city, country_id);

asianCountryys = filter countryys by region matches 'Asia';

joined = join customers by country_id, asianCountryys by country_id;

grouped = group joined by country_name;

agged = foreach grouped generate group, COUNT(joined.customers::cust_id);

morethan500cust = filter agged by $1 > 500;

ordered = order morethan500cust by $1 desc;

dump ordered;

**SQL or Hive QL**

```
SELECT country_name, COUNT(cust_id) AS cust_count
FROM countries co
JOIN customers cu
ON (co.country_id=cu.country_id)
WHERE country_region='Asia'
GROUP BY country_name
HAVING COUNT(cust_id)>500
ORDER BY cust_count DESC
```
Considerations

• Hadoop systems are now widely deployed in commercial, production environments and
  – No longer “knowable” workloads. Mixed workloads are introduced by user community and often cannot be analyzed until they are running.
  – Typical to see “word-of-mouth” sharing of resources
  – Hadoop has over 175 tunable parameters. Tuning is often “gut feel”. And is also highly workload dependent.
Considerations

• Performance issues are common.
  – Default scheduler is first come first serve, so a submitted job is run as soon as possible
    • Overloaded systems and disks
    • Under-committed systems and disks
    • Longer than desired job run times
      – Sometimes just waiting for the last piece of the job to finish
Considerations

• Algorithms really matter
  – Data partitioning strategies
  – Map, Combiner, and Reduce complexities
  – Even simple aggregation tasks can be pretty complex
    – Compare and contrast to PLINQ
    – Greenplum, Vertica, IBM, MapR, et al

• Criticisms leveled by DeWitt and Stonebraker
Considerations

• Testing matters
  – Systems such as this are inherently difficult to test
    • Large complex job streams
    • Test data and results
    • Load testing, Failover testing, system stability
  – Tools have not yet caught up. There are log files everywhere!

– I love sed, awk and grep, really I do!

• It’s like it’s 1975 all over again
Considerations

• So, how big of a problem is security?
  – This is a major criticism. Is it the
    in the room?
  – Added-on “after-the-fact”
  – Work-arounds abound, but.....??????
Considerations

• The same old song and dance
  – Commodity versus proprietary
Considerations

• The same old song and dance
  – Commodity versus proprietary
  – Hardware, software and labor expense tradeoffs
Considerations

• The same old song and dance
  – Commodity versus proprietary
  – Hardware, software and labor expense tradeoffs
  – Rapidly evolving (mutating?) software standards
    • “Building tomorrow’s legacy systems...today!”™
Considerations

• The same old song and dance
  – Commodity versus proprietary
  – Hardware, software and labor expense tradeoffs
  – Rapidly evolving (mutating?) software standards
    • “Building tomorrow’s legacy systems...today!” ™

• Or, as Stephen Colbert says, are we “Building a better tomorrow... tomorrow.” ????
Considerations

• Big Data solutions using Hadoop are Cloud-ready
  – Used for Development pooling and sandboxing
  – Used for testing – but beware of load testing!
  – Used ultimately for overflow and peak to average usage surges.
Considerations

• Big Data solutions using Hadoop are Cloud-ready
  – Used for Development pooling and sandboxing
  – Used for testing – but beware of load testing!
  – Used ultimately for overflow and peak to average usage surges.

But, is virtualization a friend or foe?
Summary

• **Big Data** has quietly become mainstream, but suddenly one cannot seem to pick up a magazine without seeing it.

• Commercial products and services vendors are springing up daily across a wide variety of markets.

• Must consider the impact of integration, performance and testing.
Questions and Answers