Strategies for Managing Large-Scale Data: Database Systems, Cloud Computing, and Hadoop

Sriram Krishnan
Viswanath Nandigam
Outline

• Traditional Database Implementations for large-scale spatial data
• Hybrid Systems – Database & Files based
• Introduction to Cloud Computing
• Introduction to the MapReduce and Hadoop
  – Distributed File System
  – Programming model
• What’s in it for me?
Data Partitioning: Shared Nothing

Data Partitioning Strategies

• **Round Robin**
  – Equal distribution across nodes by data volume

• **Hash**
  – All data with the same key value go to the same node

• **Range**
  – All data within a range of values go to the same node
LIDAR Database (Hardware/Software)

- **DB2 Enterprise Server Edition (Version 8.2 fixpack 14)**
  - Data Partitioning Feature (DPF)
- **DB2 Spatial Extender (Version 8.2)**
  - Advanced spatial data types
  - Advanced spatial data types functions and features

- **GEON LIDAR Cluster Hardware**
  - 8 x (dual core Intel Xeon 3.0 GHz processors with 8GB of memory and 750 GB local disk on each)
  - Four 3TB disk arrays - 1.5 TB local to each node
  - Total Disk ~ 18 TB
Sample LIDAR ASCII Dataset

Northern San Andreas Fault Data

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>X</th>
<th>Y</th>
<th>Elevation</th>
<th>Return Number</th>
<th>Number of Returns</th>
<th>Off Nadir Angle</th>
<th>Return Intensity</th>
<th>Classification Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1205</td>
<td>148544.74364</td>
<td>6135780.64</td>
<td>2074881.21</td>
<td>1557.35</td>
<td>1</td>
<td>5</td>
<td>5.48</td>
<td>14</td>
<td>V</td>
</tr>
<tr>
<td>1205</td>
<td>148544.76241</td>
<td>6135770.33</td>
<td>2074880.95</td>
<td>1526.20</td>
<td>2</td>
<td>6</td>
<td>5.45</td>
<td>0</td>
<td>G</td>
</tr>
<tr>
<td>1205</td>
<td>148544.76246</td>
<td>6135775.01</td>
<td>2074884.30</td>
<td>1529.68</td>
<td>1</td>
<td>5</td>
<td>5.47</td>
<td>15</td>
<td>G</td>
</tr>
<tr>
<td>1205</td>
<td>148544.78122</td>
<td>6135765.79</td>
<td>2074881.62</td>
<td>1524.31</td>
<td>1</td>
<td>5</td>
<td>5.44</td>
<td>80</td>
<td>B</td>
</tr>
</tbody>
</table>

Column 1: Date - Day or week of acquisition
Column 2: Time - GPS time stamp uniquely identifying laser pulse time
Column 3 and 4: X and Y
Column 5: Elevation.
Column 6: Return number - Return number of this return.
Column 7: Number of Returns - Number of returns for this pulse.
Column 8: Off Nadir Angle - Angle between nadir and transmitted pulse
Column 9: Return Intensity - Intensity of return pulse
Column 10: Classification Code - Classification of return where:
  B-Blunder;
  G-Ground or water;
  V-Vegetation;
  S-Building/Structure;
  N-Not ground or water - Could be V or S
DB2 Spatial Indexes

- Grid index that is optimized for two dimensional data
- Logical square grids
DB2 Spatial Extender

Spatial conversions using DB2 Spatial Extender Functions
Convert to ST_Point(X,Y) using spatial projection information
NAD_1983_StatePlane_California_II_FIPS_0402_Feet

Spatial Query
Get all points within a defined spatial envelope
EnvelopesIntersect
(gr ST_POINT, g2 ST_POINT)
Database Setup Process

DB2 UDB

NSAF Objects Schema

NSAF.q39123b63

x,y,z,...Classification | Geometry (Spatial)

LiDAR Meta Data Table

Table Name, max(X,Y), min(X,Y), No. of rows, quadrant Name, Projection Plane

RAINIER Objects Schema

ECSZ Objects Schema

NSAF Dataset
q39123b63/
q39123b6301.txt
q39123b6302.txt
q39123b6303.txt

Generate ST.POINT(X,Y) using spatial projection information (NAD 1983 StatePlane California II FIPS 0402 Feet)
LIDAR Database Cluster
LIDAR ASCII Database Approach: Experiences

• **Pros**
  – Simple SQL-based querying of data
  – Robust production-quality software/hardware stack
  – High performance access to data

• **Cons**
  – Increase in DB size due to indexing
  – Extract, Transform, Load (ETL) overhead
  – Fault Tolerance
  – Scalability
  – Costs (licensing, hardware)
DB Approach Issues: Size

• Size of spatial objects in database
  – ASCII File
    • X, Y, …..attributes
  – Data + ST_POINT + Spatial/Other Indexes
  – ~ 7 X Increase
  – GARLOCK Data:
    Raw file: 113 GB
    Size in DB: 852 GB
DB Approach Issues: Extract, transform, and load (ETL) Overhead

- Preprocess raw LIDAR data files (10%)
- DDL, loading process (25%)
  Tables, bufferpools, tables, indexes, load and spatial generation scripts
- Define Spatial Reference System and spatial columns (5%)
  ```
  db2se create_srs MYSRS -srsName UTM_10N -srsId 1011 -
  xOffset 0 -yOffset 0 -coordsysName
  WGS_1984_UTM_ZONE_10N;
  ```
- Populate the spatial column (35%)
  ```
  db2 update myTable set GEOMETRY = db2gse.GsePoint(X, Y,
  1011, ... );
  ```
- Create spatial and other indexes (20%)
  ```
  db2 create index MYSINDEX on myTable(GEOMETRY) extend
  using db2gse.spatial_index(1000,0,0)
  ```
- Optimize tables and update meta data table (5%)
DB Approach Issues: Fault Tolerance
DB Approach Issues: Scalability

- Single Partition Group
  - Single Node TableSpace
- Small Partition Group
  - Eastern California Sheer Zone TS
- Medium Partition Group
  - West Rainier Seismic Zone TS
- Large Partition Group
  - GeoEarthScope Northern California LIDAR Project TS

+ Nodes
LAS Binary File Format

- ASCII – Slow reads/Large size/LIDAR Specific information is lost.
- The ASPRS LAS binary format is the industry standard for LIDAR data exchange.
- http://www.asprs.org/a/society/committees/standards/lidar_exchange_format.html
Tools for LAS

• libLAS
  – C/C++ library
  – http://liblas.org/

• LAStools
  – http://www.cs.unc.edu/~isenburg/lastools/
Hybrid Implementation
Database + Files

• The point cloud data files are stored on the file server in binary LAS format
• The metadata for the LAS files is stored in a relational database for faster querying.
LIDAR LAS System

LAS Server

LAS Module

Get Intersecting LAS Files

Dataset Selection

LAS Metadata Repository

Meta Table

NAPA Dataset Metadata

Initial Load Dataset Metadata

LAS Dataset 1

LAS Files

LAS Dataset 2

LAS Files

NAPA Dataset

libLAS libraries to query LAS files subset
Hybrid Approach Experiences

Issues Addressed:

– **Increase in DB size due to indexing**
  • Size is no longer an issue.

– **Extract, Transform, Load (ETL) overhead**
  • ETL is significantly faster

– Performance/Cost ratio is much better.
Hybrid Approach

Issues not addressed:
– Fault Tolerance
– Scalability
– Costs
Cloud Computing

- Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
On-demand Self Service

• A consumer can unilaterally provision computing capabilities, such as server time and network storage
  – As needed
  – Automatically
  – Without requiring human interaction from service provider
Ubiquitous Network Access

• Capabilities are available over the network
  – Typically through Web service APIs

• Accessed through standard mechanisms
  – Heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
  – Typically via light-weight Web service APIs
Rapid Elasticity

• Capabilities can be rapidly and elastically provisioned to quickly scale up
  – And rapidly released to quickly scale down

• To the consumer, the capabilities available for provisioning often appear to be infinite

• Can be purchased in any quantity at any time
Measured Servicing

• Automatic control and optimization of resource use by leveraging a metering capability
  – At the level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts)

• Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service
Deployment Models

• Private Cloud
  – Operated solely for an organization
  – May be managed by the organization or a third party and may exist on premise or off premise

• Community Cloud
  – Shared by several organizations and supports a specific community that has shared concerns
  – May be managed by the organizations or a third party and may exist on premise or off premise
Deployment Models

• Public Cloud
  – Made available to the general public or a large industry group
  – Owned by an organization selling cloud services

• Hybrid Cloud
  – A composition of two or more clouds (private, community, or public)
  – Bound together by standardized or proprietary technology that enables data and application portability
Delivery Models

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)
Infrastructure as a Service (IaaS)

• The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources

• Consumer is able to deploy and run arbitrary software, which can include operating systems and applications

• The consumer does not manage or control the underlying cloud infrastructure
  – But has control over operating systems, storage, deployed applications, and possibly select networking components (e.g., firewalls, load balancers)

• Examples: Amazon EC2 (compute), S3 (storage)
Platform as a Service (PaaS)

• The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created applications using programming languages and tools supported by the provider (e.g., Java, Python, .Net)

• The consumer does not manage or control the underlying cloud infrastructure, network, servers, operating systems, or storage
  – But the consumer has control over the deployed applications and possibly application hosting environment configurations

• Examples: Google AppEngine, Amazon Elastic MapReduce (EMR), Microsoft Azure
Software as a Service (SaaS)

• The capability provided to the consumer is to use the provider’s applications running on a cloud infrastructure

• Accessible from various client devices through a thin client interface such as a Web browser (e.g., web-based email)

• The consumer does not manage or control the underlying cloud infrastructure
  – With the possible exception of limited user-specific application configuration settings

• Examples: Google Maps, Facebook, Animoto
Cloud Computing Costs*

Cloud offerings currently are most attractive for small and medium-sized enterprises

Clouds are very cost effective for SMEs ...

... and most customers of clouds are small businesses

* Source: McKinsey & Co
Observations on Cloud Computing

• Enterprises incur no infrastructure capital costs, just operational costs on a pay-per-use basis
  – Elimination of up-front costs
• Architecture specifics are abstracted
• Capacity can be scaled up or down dynamically, and immediately
  – Illusion of infinite resources, on-demand
• The underlying hardware can be anywhere geographically
When to use Clouds?

- **Do your workloads change drastically from time to time?**
  - Clouds provide excellent scaling up and down
  - Pay as you go, only for the resources you use
- **Do you need a modest amount or a large amount of compute/data resources?**
  - Prices do add up as your requirements grow
- **What business are you in? Do you care to run your own resources/data center?**
  - Off-load resource management to cloud providers
  - Focus on your own key expertise
- **Are you the impatient type 😊?**
  - Do you want access to resources NOW?
Other Obstacles*

*Source: “Above the Clouds: A Berkeley View of Cloud Computing”

- **Service Availability**
  - Services on the internet do go down

- **Vendor Lock-In**
  - Customers can’t easily extract and transfer their data or programs from one site to another

- **Data Confidentiality and Auditability**
  - Inherent trust issues in migrating one’s data to the cloud

- **Data Transfer Bottlenecks**
  - Getting large amounts of data (>TB) in and out of clouds extremely time consuming

- **Performance Unpredictability**
  - Since multiple virtual machines share CPU, Memory and I/O
Other Obstacles

• Scalable storage
  – Elasticity is more straightforward with respect to computation, not storage
• Bugs in large distributed systems
  – Debugging of large scale systems extremely complex
• Scaling quickly
  – Automatically scale applications in response to load
• Reputation Fate Sharing
  – One customer’s bad behavior can affect the reputation of the cloud as a whole
• Software Licensing
  – May restrict the computers on which software may be run
Comparison: Cloud Versus Grid

<table>
<thead>
<tr>
<th></th>
<th>Cloud</th>
<th>Grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Type</td>
<td>Business, Web</td>
<td>HPC</td>
</tr>
<tr>
<td>Number of Users</td>
<td>Large number of concurrent users</td>
<td>Smaller numbers of highly specialized</td>
</tr>
<tr>
<td>Access Type</td>
<td>Non-exclusive access to smaller number of resources at a time</td>
<td>Typically exclusive access, to large number of resources</td>
</tr>
<tr>
<td>Access Mechanisms</td>
<td>On-demand elastic access typically via Web service APIs</td>
<td>Batch queuing systems for job submission</td>
</tr>
</tbody>
</table>
Data Processing Techniques on the Cloud

- Cloud computing necessitates a different approach for large scale data processing
  - Use of non-specialized commodity hardware and low latency interconnect between nodes
  - Not suitable for implementing traditional databases or HPC systems which require specialized hardware configurations
  - Traditional databases also not known to scale very well to thousands of nodes which may be available on the cloud
Traditional HPC Architectures

Shared-nothing (MapReduce-style) Architectures
The MapReduce Paradigm

- Programming environment for very large scale data processing on commodity resources
  - Originally developed by Google
- Managing task executions and data transfers in a “shared-nothing” environment
  - Distributed data repository (“file system”): Google File System (GFS)
    - Round-robin partitioning of data
  - MapReduce: Infrastructure to support data scatter / gather
    - Pushing the computation to the data
Hadoop

• Open Source Implementation of Google’s MapReduce ecosystem
  – Hadoop Distributed File System (HDFS): Based on the Google File System
  – Hadoop MapReduce: Based on Google’s MapReduce

• http://hadoop.apache.org/
Distributed “File System”
HDFS Architecture

*Source: [http://hadoop.apache.org/common/docs/current/hdfs_design.html](http://hadoop.apache.org/common/docs/current/hdfs_design.html)
The MapReduce Paradigm

- The computation takes a set of *input key/value pairs*, and
- Produces a set of *output key/value pairs*.
- The MapReduce library expresses the computation as two functions
  - Map and Reduce
- All data resides in a distributed file system
MapReduce: Types

- **map** \((k_1,v_1) \rightarrow \text{list}(k_2,v_2)\)
  - Map function transforms input (key, value) pairs into a list of intermediate (key, value) pairs

- **reduce** \((k_2,\text{list}(v_2)) \rightarrow \text{list}(v_2)\)
  - Reduce function transforms a key and a set of its values to a final list of values (outputs)
Word Count in MapReduce

```java
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1");

reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result))
```
MapReduce Execution*

*Source: http://labs.google.com/papers/mapreduce.html
MapReduce: Observations

- **Data can be processed “in-place”**
  - No need to export data from database for downstream processing

- **Easy to use – complexity is hidden by the library**

- **A large variety of problems are easily expressible in this model**
  - Especially ones that are embarrassing parallel, and deal with processing of large amounts of data

- **Caveat: Not all algorithms lend themselves very well to be implemented using this model**

- **Restricting the programming model makes it easy to parallelize and deal with failures**
Hadoop: What is it good for?

- **Scalability**
  - Can (theoretically) grow or shrink the size of Hadoop clusters depending upon workloads?
  - Clusters can scale up to 1000s of nodes

- **Fault tolerance**
  - Can handle failures to worker nodes seamlessly
  - Current version still vulnerable to master failures (currently being addressed)

- **Cost**
  - No need for specialized hardware – so good price/performance

- **Batch processing**
  - Processing of large amounts of data, in a non-interactive fashion
Hadoop: What is it not good for?

• Not all algorithms are easily expressible in MapReduce
  – Codes that are tightly coupled, and need frequent communication are not a good fit
  – Graph algorithms are not a natural fit (requires multiple iterations in MapReduce, which are wasteful)

• Certain database operations are less efficient in MapReduce
  – Hadoop/MapReduce doesn’t have a concept of indexes – so index-based operations will be much slower
  – Joins are also more efficient in databases
    • Databases have robust optimization techniques – the programmer is the optimizer in Hadoop

• High latency for interactive workloads
  – Hadoop is not suitable for such workloads
Sample Performance Comparison*

Data Load Times – 1TB/Cluster

Execution time – Distributed Grep

*“A Comparison of Approaches to Large-scale Data Analysis”, Pavlo et al, SIGMOD 09
MapReduce Examples in Scientific Computing
Bioinformatics: Crossbow*


Credit:
Geosciences: Digital Elevation Models (DEM)

Full-featured DEM

Bare earth DEM
Geosciences: Hadoop-based DEM Generation*

Where/How to Run Hadoop Code

• **Dedicated Hadoop Cluster**
  – Set of commodity nodes configured to run Hadoop in distributed mode

• **On the cloud (e.g. Amazon Elastic MapReduce)**
  – Variable number of nodes uses Amazon’s Elastic MapReduce

• **On HPC resources**
  – Need scripts to configure Hadoop on-demand via traditional batch systems
  – HOD: [http://hadoop.apache.org/common/docs/r0.18.3/hod.html](http://hadoop.apache.org/common/docs/r0.18.3/hod.html)
Amazon Elastic MapReduce

Creating a job flow to process your data using Amazon Elastic MapReduce is simple and quick. Let's begin by giving your job flow a name and selecting its type. If you don't already have an application you'd like to run on Amazon Elastic MapReduce, samples are available to help you get started.

**Job Flow Name**: My Job Flow

The name can be anything you like and doesn't need to be unique. It's a good idea to name the job flow something descriptive.

**Type**: Streaming

A Streaming job flow allows you to write single-step mapper and reducer functions in a language other than Java.

- Custom Jar (advanced)
  A custom jar on the other hand gives you more complete control over the function of Hadoop but must be a compiled java program. Amazon Elastic MapReduce supports custom jars developed for Hadoop 0.18.3.

- Pig Program
  Pig is a SQL-like language built on top of Hadoop. This option allows you to define a job flow that runs a Pig script, or set up a job flow that can be used interactively via SSH to run Pig commands.

**Sample Applications**

Select a sample application and click Continue. Subsequent forms will be filled with the necessary data to create a sample Job Flow.

- Word Count (Streaming)
  Word count is a Python application that counts occurrences of each word in provided documents. Learn more and view license

* Required field
Amazon Elastic MapReduce

Create a New Job Flow

Define Job Flow  Specify Parameters  Configure EC2 Instances  Review

Specify Mapper and Reducer functions to run within the Job Flow. The mapper and reducers may be either (i) class names referring to a mapper or reducer classes in Hadoop or (ii) locations in Amazon S3. (Click Here for a list of available tools to help you upload and download files from Amazon S3.) The format for specifying a location in Amazon S3 is bucket_name/path_name. The location should point to an executable program, for example a python program. Extra arguments are passed to the Hadoop streaming program and can specify things such as additional files to be loaded into the distributed cache.

Input Location*: elasticmapreduce/samples/similarity/lastfm/input
The URL of the Amazon S3 Bucket that contains the input files.
Output Location*: <yourbucket>/lastfm/output/user-counts
The URL of the Amazon S3 Bucket to store output files.
Mapper*: elasticmapreduce/samples/similarity/user_count_mapper.py
The name of the mapper executable located in the Input Location.
Reducer*: aggregate
The name of the reducer executable located in the Input Location.
Extra Args: 

* Required field
Amazon Elastic MapReduce

Create a New Job Flow

Enter the number and type of EC2 instances you'd like to run your job flow on.

Number of Instances*: 4
The number of EC2 instances to run in your Hadoop cluster.

Type of Instance*: Small (m1.small)
The type of EC2 instances to run in your Hadoop cluster (learn more about instance types).

Show advanced options

Back Continue
Amazon Elastic MapReduce

Create a New Job Flow

Please review the details of your job flow and click "Create Job Flow" when you are ready to launch your Hadoop Cluster.

Job Flow Name: User Count (Streaming)
Type: Streaming

Input Location: s3n://elasticmapreduce/samples/similarity/lastfm/input
Output Location: s3n://aws-hadoop/lastfm/output/user-counts
Mapper: s3n://elasticmapreduce/samples/similarity/user_count_mapper.py
Reducer: s3n://elasticmapreduce/samples/similarity/user_count_reducer.py
Extra Args: aggregate

Number of Instances: 4
Type of Instance: m1.small
Amazon S3 Log Path: 
Amazon EC2 Key Pair:

Note: Once you click "Create Job Flow," instances will be launched and you will be charged accordingly.
More Reading

  - Use of custom scripts as mappers and reducers
  - No Java code required
- **Disco (MapReduce in Python)**: [http://discoproject.org/](http://discoproject.org/)
Questions?

• Feel free to email me (sriram@sdsc.edu)
Appendix
Location Independent Resource Pooling

- The provider’s computing resources are pooled to serve all consumers using a multi-tenant model
  - Different physical and virtual resources dynamically assigned and reassigned according to consumer demand
- The customer generally has no control or knowledge over the exact location of the provided resources
  - But may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter)
- Examples of resources include storage, processing, memory, network bandwidth, and virtual machines
IaaS: Amazon Web Services (AWS)

- **Amazon Elastic Compute Cloud (EC2)**
  - A web service that provides resizable compute capacity in the cloud.
  - Configure an Amazon Machine Instance (AMI) and load it into the Amazon EC2 service
  - Quickly scale capacity, both up and down, as your computing requirements change

- **Amazon Simple Storage Service (S3)**
  - A simple web services interface that can be used to store and retrieve large amounts of data, at any time, from anywhere on the web
  - It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites
PaaS: Google AppEngine

*Credit: http://rdn-consulting.com/blog/2009/02/07/exploring-cloud-computing-development/