Forest Aboveground Biomass and Carbon Mapping With Computational Cloud

by

Aimin Guan

BSc, University of Victoria, 2006

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

© Aimin Guan, 2017

University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.
Supervisory Committee

Forest Aboveground Biomass and Carbon Mapping With Computational Cloud

by

Aimin Guan
BSc, University of Victoria, 2006

Supervisory Committee

Dr. David G. Goodenough, (Department of Computer Science)
Supervisor

Dr. Wendy Myrvold, (Department of Computer Science)
Co-Supervisor

Dr. Olaf Niemann (Department of Geography)
Outside Member
Abstract

In the last decade, advances in sensor and computing technology are revolutionary. The latest-generation of hyperspectral and synthetic aperture radar (SAR) instruments have increased their spectral, spatial, and temporal resolution. Consequently, the data sets collected are increasing rapidly in size and frequency of acquisition. Remote sensing applications are requiring more computing resources for data analysis. High performance computing (HPC) infrastructure such as clusters, distributed networks, grids, clouds and specialized hardware components, have been used to disseminate large volumes of remote sensing data and to accelerate the computational speed in processing raw images and extracting information from remote sensing data. In previous research we have shown that we can improve computational efficiency of a hyperspectral image denoising algorithm by parallelizing the algorithm utilizing a distributed computing grid. In recent years, computational cloud technology is emerging, bringing more flexibility and simplicity for data processing. Hadoop MapReduce is a software framework for distributed commodity computing clusters, allowing parallel processing of massive datasets. In this project, we implement a software application to map forest aboveground biomass (AGB) with normalized difference vegetation indices (NDVI) using Landsat Thematic Mapper’s bands 4 and 5 (ND45). We present observations and experimental results on the performance and the algorithmic complexity of the implementation. There are three research questions answered in this thesis, as follows. 1) How do we implement remote sensing algorithms, such as forest AGB mapping, in a computer cloud environment? 2) What are the requirements to implement distributed processing of remote sensing images using the cloud programming model? 3) What is the performance increase for large area remote sensing image processing in a cloud environment?
# Table of Contents

**Supervisory Committee** ......................................................................................... iii
**Abstract** .................................................................................................................. iii
**Table of Contents** ..................................................................................................... iv
**List of Figures** ......................................................................................................... vi
**List of Tables** .......................................................................................................... viii
**Glossary** .................................................................................................................. ix
**Acknowledgments** ................................................................................................. xi
**Dedication** ............................................................................................................... xii

## Chapter 1 Introduction

1.1 Aboveground Biomass and Kyoto Protocol ............................................................ 1
1.2 Biomass Estimation Methods .................................................................................. 3
1.3 Remote Sensing ....................................................................................................... 5
   1.3.1 Multispectral Sensor ......................................................................................... 6
   1.3.2 Hyperspectral Remote Sensing ........................................................................ 10
   1.3.3 Radar Remote Sensing ..................................................................................... 13
1.4 High Performance Computing and Remote Sensing .............................................. 15
1.5 Research Objective and Research Questions ......................................................... 20

## Chapter 2 Aboveground Biomass Estimation Methods

2.1 National Forest Inventory ....................................................................................... 21
2.2 Calculating Large Area AGB with Remote Sensing Images .................................... 23
   2.2.1 Vegetation Index ............................................................................................. 23
   2.2.2 ND45 .............................................................................................................. 24
   2.2.3 Remote Sensing Image with Inventory Data .................................................. 26

## Chapter 3 Remote Sensing Image Processing and Calibration

3.1 Ground Measurement ............................................................................................ 28
3.2 Geometric Correction ........................................................................................... 29
3.3 Radiometric Correction ......................................................................................... 29
3.4 Atmospheric Correction ......................................................................................... 30
3.5 Classification Methods ......................................................................................... 30
   3.5.1 K-means Clustering ......................................................................................... 31
   3.5.2 Maximum Likelihood Classification ............................................................... 33
   3.5.3 Sample Size ................................................................................................... 36
   3.5.4 Support Vector Machine ................................................................................. 37
3.6 Accuracy Assessment ............................................................................................ 40
3.7 Principal Component Analysis .............................................................................. 41
List of Figures

Figure 1. Diagram of the forest carbon cycle showing the movement of carbon among land, atmosphere, and ocean. Yellow numbers represent fluxes to do with natural processes, whereas red numbers represent human contributions (in gigatons of carbon produced, per year). White numbers indicate quantities of stored carbon. This picture is taken from [5]................................. 2

Figure 2. Representation of National Forest Inventory 20km by 20km ground plot grid across Canada[12]. In the legend, the circle represents ground plots; the squares represent 2 km by 2 km photo plots. Ten percent of the ground plots are randomly selected by sampling and established as 2 km by 2 km photo plots. .......................................................................................................................... 4

Figure 3. A conceptual structure of an optical scanning system that use a linear array of detectors located at the focal plane of the image formed by lens systems, which are "pushed" along in the flight track direction [1]........................................................................................................................................ 5

Figure 4. Landsat mission timeline[21]. ........................................................................................................................................ 5

Figure 5. Canada’s Forest Cover, a Landsat Frame Path 47, Row 26 (July 30, 2000) capturing Victoria and Vancouver BC, and the associated data collection path. ................................................. 6

Figure 6. EOSD land cover map and class legends: EOSD Land cover map Victoria and Vancouver, BC (left) and EOSD land cover classes Legend used in each tile for the year 2000 (right). ........................................................................................................................................ 8

Figure 7. Left: AVIRIS 4m image acquired in 2001 with 3682 samples, 5874 lines, and 179 bands. The color image was formed using bands red: 128, green: 44, and blue: 26. Right: the z dimension is the reflectance information (wavelength from 380 nm to 2400 nm). ................. 10

Figure 9. Individual tree biomass contains four components: stem wood, stem bark, branches and foliage. The total of the biomass is the sum of the total volumes of the four components.... 22

Figure 9. Three discrete components: merchantable trees, non-merchantable trees, and sapling trees within a ground plot in NFI plot. The plot type includes main plots and sub-plots........ 23

Figure 10. ND45 algorithm flow chart ........................................................................................................................................ 26

Figure 11. The K-means Clustering Algorithm ..................................................................................................................................... 32

Figure 12. EOSD classification procedure flow. ..................................................................................................................................... 33

Figure 13. A general strategy for supervised classification ..................................................................................................................................... 37

Figure 14. The separating hyperplane of classes "-1" and "+1" ........................................................................................................... 38
Figure 15. SVM kernel function $\varphi$ ................................................................. 39
Figure 16. Non-linear Denoising Algorithm: UML Class Diagram................................. 51
Figure 17. Flowchart of the shell script executing on worker node. This controls de-noising program execution. ........................................................................................................ 52
Figure 18. Hyperspectral denoising program: worker nodes flowchart. ............................ 53
Figure 19. Example of the denoising process for an AVIRIS image................................. 54
Figure 20. Total Processing Time vs. the Number of Sub-images ..................................... 55
Figure 21. The “ResourceManager” (RM) has two components: the Scheduler and Application Master (AM) .................................................................................................................. 58
Figure 22. Hadoop Distributed File System (HDFS) Architecture [70]. .......................... 60
Figure 23. HDFS file system checking command. The output shows the file with its locations information.................................................................................................................. 61
Figure 24. Data Flow for Hadoop MapReduce Processing Model of ND45 ....................... 63
Figure 25. Landsat images and EOSD land cover classification map. ............................... 69
Figure 26. “Map” job: Carbon generation processing flow ............................................. 71
Figure 27. (a) The above ground carbon map over the map area. (b) The legend of carbon map. 72
Figure 28. Parallelization results showed the increasing performance when using multiple VMs Hadoop MapReduce cluster to process AGB. ................................................................. 73
List of Tables

Table 1. The Landsat Missions by sensor type, wavelength bands, and spatial resolution[21] ..... 7
Table 2. EOSD land cover classes used in each tile as determined in the year 2000. ............... 9
Table 3. Hyperspectral Sensors and Spectral Specifications.......................................................... 11
Table 4. Radar Imaging sensors, bands, and polarizations............................................................ 14
Table 5. Confusion Matrix.................................................................................................................. 40
Table 6. Master / Slave VM Configuration ....................................................................................... 67
Table 7. Software Configuration for Cluster Compute Nodes (Virtual Machines)......................... 68
Table 8. Landsat 7 ETM+ data acquisitions ..................................................................................... 69
Table 9. Run time for carbon mapping using Landsat imagery and ND45 model. ....................... 72
Glossary

AGB  Forest Above Ground Biomass
AISA  Airborne Hyperspectral Imaging Systems
AM  Application Master
ASD  Analytical Spectral Devices Inc.
AVIRIS  Airborne Visible Infrared Imaging Spectrometer
CFS  Canadian Forest Service
COTS  Commercial off-the-Shelf
DAG  Directed Acyclic Graph
DBH  Diameter at Breast Height
EOSD  Earth Observation for Sustainable Development of Forests
ESA  European Space Agency
GCPs  Ground Control Points
GDAL  Geospatial Data Abstraction Library
GHG  Greenhouse Gas
GT4  Globus Toolkit Version 4
GVWD  Great Victoria Watershed District
HDFS  Hadoop distributed file system
HPC  High Performance Computing
IaaS  Infrastructure-as-a-service
IPCC  Intergovernmental Panel on Climate Change
LAI  Leaf Area Index
LULUCF  Land-use, Land-Use Change and Forestry
MERIS  MEdium Resolution Imaging Spectrometer
MIMD  Multiple Instructions Multiple Data
MLC  Maximum Likelihood Classification
MNF  Minimum Noise Fraction
MODIS  Moderate Resolution Imaging Spectroradiometer
NDVI  Normalized Difference Vegetation Index
NFCMARS  National Forest Carbon Monitoring, Accounting and Reporting System
OGC  Open Geospatial Consortium
PCA  Principal Component Analysis
ppm  Parts per million
RM  Resource Manager
SaaS  Software-as-a-service
SAFORAH  System of Agents for Forest Observation Research with Advanced Hierarchies
SIMD  Single Instruction Multiple DataStream
SNR  Signal to Noise Ratio
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SWIG</td>
<td>Simplified Wrapper and Interface Generator</td>
</tr>
<tr>
<td>TISEAN</td>
<td>Time Series Analysis libraries</td>
</tr>
<tr>
<td>ToA</td>
<td>Top of Atmosphere</td>
</tr>
</tbody>
</table>
Acknowledgments

Firstly, I would like to acknowledge the encouragement, help, and support of my supervisor, Dr. David Goodenough. I am also much obliged to my co-supervisor Dr. Wendy Myrvold and to my outside member Dr. Olaf Niemann: thank you for serving on my committee. I would also like to thank Hao Chen for his assistance with remote sensing image processing, and computing system configurations. I appreciate my colleagues in the National Forest Information System (NFIS) group for allowing me to work flexible hours while finishing my research. I am also thankful to the faculty and staff at the University of Victoria (UVic) and at the Pacific Forestry Centre (PFC), where I conducted much of my research. Acknowledgements are also given to the West Grid of Compute Canada and the Open Science Data Cloud for providing me access to their computational cloud facilities.

Programs and scripts for Above Ground Biomass (AGB) estimation written by Piper Gordon, Hao Chen, and Dr. Belaid Moa at the Department of Computer Science, University of Victoria, Remote Sensing Software Engineering Lab were used in the project. Some aspects of these programs were integrated into our parallel processing system.
Dedication

This work is dedicated to my Family, especially to my husband Jianxiang Zhai, for his support and understanding.
Chapter 1  
Introduction

1.1 Aboveground Biomass and Kyoto Protocol

*Forest biomass* is the organic material within forested areas. Forest aboveground biomass (AGB) information supports bio-energy initiatives, helps estimate timber quantity and forest carbon sequestration, and addresses many other informational needs. Determination of forest biomass is also highly relevant for the global climate change issue, especially in relation to studying the carbon cycle and greenhouse gas (GHG) emissions[2].

Greenhouse gases absorb optical radiation and block radiation in the thermal infrared range: this process is called the *greenhouse effect*. The greenhouse effect keeps the earth’s surface warm. Over the 20th century atmospheric concentrations of key greenhouse gases have increased, largely due to human activity [2]. In the last 800,000 years, the global average CO$_2$ concentration fluctuated between about 180 ppm during ice ages to 280 ppm during interglacial warm periods [3] Measurements made by National Oceanic and Atmospheric Administration (NOAA) [3] indicate that the rate of change of CO$_2$ concentration increased from about 0.7 parts per million (ppm) per year in the late 1950s to 2.1 ppm per year during the last decade [3]. The rate of change of global average CO$_2$ concentration is much more pronounced today than it was at the end of the last ice age. In May 2015 the daily mean atmospheric concentration of carbon dioxide at Mauna Loa, Hawaii (the primary global benchmark site) surpassed 400 ppm for the first time since measurements began there in 1958 [3]. Figure 1 shows the natural carbon cycle among humans, forests, soil, oceans, and atmosphere.

According to the 2001 Intergovernmental Panel on Climate Change (IPCC) Third Assessment Report, carbon dioxide (CO$_2$) produced from human activities is at least 60% responsible for climate change at present. The conclusions of Houghton et al. [4] are that the burning of fossil fuels and the consequences of land-use change are primary factors in CO$_2$
production. As a result, since the late 19th century, global surface temperatures have increased by around 0.4°C to 0.8°C.

Figure 1. Diagram of the forest carbon cycle showing the movement of carbon among land, atmosphere, and ocean. Yellow numbers represent fluxes to do with natural processes, whereas red numbers represent human contributions (in gigatons of carbon produced, per year). White numbers indicate quantities of stored carbon. This picture is taken from [5].

Forests absorb carbon dioxide from the atmosphere, storing carbon in biomass via photosynthesis, which releases oxygen into the atmosphere. Forest biomass can store carbon for decades (until the material decays). Forests act as sources or sinks for carbon at different times [6]. The Intergovernmental Panel on Climate Change (IPCC) report shows that the net carbon uptake by terrestrial ecosystems ranges from less than 1.0 pg (petagram) to as much as 2.6 pg of carbon per year for the 1990s [7]. Pan et al.[8] used data from forest inventories and long-term ecosystem carbon studies to estimate forest sinks, concluding that the world’s forests are a large and persistent carbon sink.
The 1992 IPCC was established as a body implementing an international treaty for cooperation between countries on climate change. The Kyoto Protocol extends the United Nations Framework Conventions on Climate Change (UNFCCC) and the ultimate objective of both treaties was to “stabilize greenhouse gas concentrations in the atmosphere at a level that will prevent dangerous human interference with the climate system” [9]. The protocol sets binding targets for the developed countries to reduce GHG emissions. The signed parties committed to keep their GHG emissions below a specified level, and agreed to make a 5 per cent reduction below the 1990 GHG emission level in the commitment period 2008-2012. Canada’s total GHG emission in 2012 was 718Mt, which represents an approximate 17% increase, relative to the same figure for 1990 (613Mt). The total GHG emission in Canada dropped significantly to 696Mt in 2009, compared to 736Mt in 2008. To estimate GHG emissions, the parties report carbon stocks during the commitment period. Carbon stocks are reported by national inventories on emissions and removals, where land-use, land-use change and forestry (LULUCF) activities represents one of six sectors identified by the IPCC [10].

1.2 Biomass Estimation Methods

Canada owns 400 millions hectares of forest (about 10% of the world’s forests) of which 348 million hectares are forestry lands [11]. Forest above ground biomass (AGB) is used to meet many informational needs: estimating the quantities of timber, supporting bio-energy initiatives, and estimating forest carbon sequestration. The most accurate field measurement of AGB is a destructive method of tree biomass measurement. This method involves harvesting of all the trees in the known area and measuring the weight of the different components (such as trunks, leaves, and branches) once they have been dried in an oven. Although the destructive method measures the biomass accurately, it is not feasible for large-scale analysis. For large-scale studies an alternative approach is to estimate the AGB using allometric equations, where forest tree data are used as input variables. Allometric
equations are models for estimating biomass created by regressing measured sample weights of biomass materials against forestry variables such as diameter at breast height (DBH) and tree height. Other forest inventory data may sometimes be involved in allometric equations.

Federal forest biomass information has largely been derived using plot-level estimates from Canada’s National Forest Inventory (NFI). The National Forest Inventory (NFI) is a program that was initiated by the Canadian federal government and provincial and territorial governments for forest monitoring. The NFI surveys are based on forest re-measurement over a national grid with 20 km by 20 km ground-plots as the permanent observational units (as in Figure 2). Ten percent of the ground plots are randomly selected for identification by photo-interpretation over 2 km by 2 km sub-areas. Plot attributes are collected by field measurement. Surrogate data such as aerial or satellite imagery are used to study plots that are not accessible for field work [12]. Plot attributes are used to estimate conversion models between merchantable volume biomass, from which above ground biomass estimates are derived [13].

![Figure 2. Representation of National Forest Inventory 20km by 20km ground plot grid across Canada](12). In the legend, the circle represents ground plots; the squares represent 2 km by 2 km photo plots. Ten percent of the ground plots are randomly selected by sampling and established as 2 km by 2 km photo plots.
Field measurements are often time consuming, labor intensive, and hard to acquire for remote or isolated areas. Inventory based measurement cannot provide the spatial distribution of forest types (conifer, deciduous or mixed) in large areas [14, 15]. The alternative of using direct methods to obtain AGB estimates for large areas is based on remote sensing data as the primary informational source for large area AGB estimation [16]. Previous research to estimate above ground forest biomass was based on data from both passive and active remote sensing imaging instruments, including (but not limited to): the multispectral Landsat TM/ETM and Advanced Land Imager (ALI), the hyperspectral AVIRIS, and the radar ASAR [17]. Goodenough et al. [17] demonstrated that remote sensing data can provide accurate forest classifications and aboveground biomass estimates valid over large scale areas.

1.3 Remote Sensing

Terrestrial Remote Sensing is the science of acquiring information about the Earth from a distance, typically using instruments deployed on aircraft or satellites. Remote sensing is very useful for obtaining forest information of remote or isolated areas. Many advanced studies using airborne and spaceborne remote sensing imaging systems, including multispectral, hyperspectral, and Synthetic Aperture Radar (SAR) systems, have been carried out, to estimate forest structure, to classify forests, and to find correlations among

Figure 3. A conceptual structure of an optical scanning system that use a linear array of detectors located at the focal plane of the image formed by lens systems, which are "pushed" along in the flight track direction [1].
forest volume, biomass, and other vegetative parameters [18-20].

### 1.3.1 Multispectral Sensor

Optical sensing systems record reflected energy separately in discrete wavelength ranges and are called multispectral sensors. Figure 3 taken from [1] shows the typical conceptual structure of an optical scanning system, in which a linear detector array is located on the focal plane for the image formed by the lens system. For example, Landsat satellites have continuously acquired space-based multispectral images of the Earth’s land surface, coastal shallows, and coral reefs since 1972. Figure 4 shows the timeline for the Landsat missions. Table 1 shows the sensor specifications for the Landsat missions, including the numbers of bands. The Landsat 1-3 satellites carried the Multispectral Scanner (MSS) instrument, whereas the Landsat 5 and 7 satellites carried the Thematic Mapper (TM) and Enhanced Thematic Mapper Plus (ETM+) instruments, respectively. The instrument on Landsat 5 and 7 is a “whisk-broom” type multispectral scanning radiometer that has visible, near-IR, and SWIR bands with 30-meter spatial resolution and one panchromatic band with 15-meter spatial resolution.

![Figure 4. Landsat mission timeline[21].](image)
Table 1. The Landsat Missions by sensor type, wavelength bands, and spatial resolution[21]

<table>
<thead>
<tr>
<th>Satellite</th>
<th>Sensor Type</th>
<th>Band</th>
<th>Wavelength (micrometers)</th>
<th>Resolution (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Landsat 1-3</td>
<td>Multispectral Scanner (MSS)</td>
<td>4</td>
<td>0.5-0.6</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0.6-0.7</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>0.7-0.8</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>0.8-1.1</td>
<td>80</td>
</tr>
<tr>
<td>Landsat 5/7</td>
<td>Thematic Mapper (TM)</td>
<td>1</td>
<td>0.45-0.52</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Enhanced Thematic Mapper Plus (ETM+)</td>
<td>2</td>
<td>0.52-0.60</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0.63-0.69</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0.77-0.90</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>1.55-1.75</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>10.40-12.50</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>2.09-2.35</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>0.52-0.90</td>
<td>15</td>
</tr>
<tr>
<td>Landsat 8</td>
<td>Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS)</td>
<td>1</td>
<td>0.43 - 0.45</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Launched February 2013</td>
<td>2</td>
<td>0.45 - 0.51</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0.53 - 0.59</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0.64 - 0.67</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0.85 - 0.88</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>1.57 - 1.65</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>2.11 - 2.29</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>0.50 - 0.68</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>1.36 - 1.38</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>10.60 - 11.19</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>11.50 - 12.51</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1 shows the sensor types and numbered image bands in terms of the associated wavelength and spatial resolution, for each Landsat mission. For Landsat 5/7 the blue-green band (wavelength range 0.45 µm to 0.52 µm) is used to distinguish soil from vegetation and also is used to distinguish deciduous tree vegetation from coniferous tree vegetation. The green band (0.52 µm to 0.60 µm) covers the reflectance peak from vegetation surfaces and emphasizes plant vigor. The red band (0.63 µm to 0.69 µm) emphasizes vegetation changes. Band 4 which measures reflected IR (0.76 µm to 0.90 µm) emphasizes biomass content and shorelines. Band 5 which also measures reflected IR (1.55 µm to 1.75 µm) is useful for determining soil and vegetation moisture content. Band 6 of Landsat 5, and Landsat 7 has a thermal wavelength range of 10.40 µm–12.50 µm and is useful for thermal mapping and soil
moisture estimation. Band 7 measures reflected IR (2.08 µm–2.35 µm) and is useful for mapping hydrothermally altered rocks associated with mineral deposits. For Landsat 7, band 8 is a panchromatic band (0.52 µm to 0.90 µm) and is useful for ‘sharpening’ multispectral images [21]. Figure 5 shows a Landsat frame: Path 47, Row 26, and was captured over Victoria and Vancouver BC area on July 30, 2000. Land cover information is essential for supporting forest monitoring and management. In Canada, the Earth Observation for Sustainable Development of Forests (EOSD) project[22] produced a fine resolution national forest cover map, using 30m Landsat TM images

Figure 5. Canada’s Forest Cover, a Landsat Frame Path 47, Row 26 (July 30, 2000) capturing Victoria and Vancouver BC, and the associated data collection path.

The EOSD project, a joint effort of Natural Resources Canada with support from the Canadian Space Agency and in collaboration with the provinces, territories and other federal agencies, used Landsat satellite data to produce a unique cross-country map of Canada’s
forested land cover. The resulting forest cover map consists of 610 map sheets, or tiles, each of which represents an area of about 15,000 square kilometers [24].

Table 2. EOSD land cover classes used in each tile as determined in the year 2000.

<table>
<thead>
<tr>
<th>No.</th>
<th>Class</th>
<th>No.</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No Data</td>
<td>13</td>
<td>Wetland - Shrub</td>
</tr>
<tr>
<td>2</td>
<td>Shadow</td>
<td>14</td>
<td>Wetland - Herb</td>
</tr>
<tr>
<td>3</td>
<td>Cloud</td>
<td>15</td>
<td>Coniferous - Dense</td>
</tr>
<tr>
<td>4</td>
<td>Snow/Ice</td>
<td>16</td>
<td>Coniferous - Open</td>
</tr>
<tr>
<td>5</td>
<td>Rock/Rubble</td>
<td>17</td>
<td>Coniferous - Sparse</td>
</tr>
<tr>
<td>6</td>
<td>Exposed Land</td>
<td>18</td>
<td>Broadleaf - Dense</td>
</tr>
<tr>
<td>7</td>
<td>Water</td>
<td>19</td>
<td>Broadleaf - Open</td>
</tr>
<tr>
<td>8</td>
<td>Shrub - Tall</td>
<td>20</td>
<td>Broadleaf - Sparse</td>
</tr>
<tr>
<td>9</td>
<td>Shrub - Low</td>
<td>21</td>
<td>Mixed Wood - Dense</td>
</tr>
<tr>
<td>10</td>
<td>Herb</td>
<td>22</td>
<td>Mixed Wood - Open</td>
</tr>
<tr>
<td>11</td>
<td>Bryoids</td>
<td>23</td>
<td>Mixed Wood - Sparse</td>
</tr>
<tr>
<td>12</td>
<td>Wetland - Treed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6 shows an EOSD land cover map and class legends. Table 2 shows the 23 land cover classes in the tiles as they existed around the year 2000. The EOSD land cover map was developed primarily to support Canada’s national and international reporting requirements for sustainable forest management, and provide information for biomass estimates and Canada's National Forest Inventory - particularly for northern forest regions where there are few ground or photo plots, and where land cover classifications have been based on limited information.
1.3.2 Hyperspectral Remote Sensing

Landsat imaging missions carry multispectral sensors recording image data in multiple (7, 8, or 11) bands, whereas hyperspectral sensors provide better resolution of surface properties by recording image data in many more and narrower spectral bands, of which there may be as many as several hundred. Hyperspectral sensors, also known as hyperspectral imaging spectrometers, collect spectral information across a continuous spectrum by dividing the spectrum into many narrow buckets (called spectral bands). Hyperspectral imaging spectrometers may be airborne or spaceborne and are able to provide accurate measurements for monitoring of changes in the oceans, in the atmosphere, and on land. MODIS, MERIS, AVIRIS, AISA Dual, and Hyperion are important hyperspectral imaging spectrometers for which the spectral specifications are exhibited in Table 3. Multi-
year data sets will help to answer questions about global land cover and land use changes, and about global climate change.

Table 3. Hyperspectral Sensors and Spectral Specifications

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Platform</th>
<th>Spectral (nm)</th>
<th>Number of bands</th>
<th>Bandwidth (nm)</th>
<th>Spatial (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODIS</td>
<td>Terra/Aqua</td>
<td>400-970</td>
<td>36</td>
<td>10-500</td>
<td>250 500 1000</td>
</tr>
<tr>
<td>MERIS</td>
<td>ENVISAT</td>
<td>412.5-900</td>
<td>15</td>
<td>3.75-20</td>
<td>260X290 (full) 1040X1160 (reduced)</td>
</tr>
<tr>
<td>Hyperion</td>
<td>EO-1</td>
<td>356 - 2577</td>
<td>220</td>
<td>10</td>
<td>30m</td>
</tr>
<tr>
<td>AVIRIS</td>
<td>Airborne</td>
<td>400 – 2450</td>
<td>224</td>
<td>10</td>
<td>20m 4m</td>
</tr>
<tr>
<td>AISA Dual</td>
<td>Airborne</td>
<td>400 – 2450</td>
<td>Up to 500</td>
<td>2.3 - 23.2</td>
<td>Various</td>
</tr>
</tbody>
</table>

MODIS (or Moderate Resolution Imaging Spectroradiometer) is a key instrument aboard the NASA Terra and Aqua satellites. MODIS observes a wide spectral range of electromagnetic energy from the visible frequencies to the near infrared frequencies, and generates 36 bands. MODIS produces images at three spatial resolutions: 250m, 500m, and 1000m and is able to map the entire globe once every two days. Landsat’s Enhanced Thematic Mapper Plus, on the other hand, reveals the Earth in finer spatial detail, but can only image a given area once every 16 days. MODIS’s high-quality daily measurements allow scientists to track changes in land cover types and land use. The recent product produced by the North American Land Change Monitoring System (NALCMS) is the 2005 Land Cover Database of North America at 250m spatial resolution, which can be used to address issues such as climate change, carbon sequestration, biodiversity loss, and changes in ecosystem structure and function.

MERIS (MEdium Resolution Imaging Spectrometer) is one of the instruments on board the European Space Agency (ESA) ENVISAT satellite launched in 2002. MERIS
measures the Earth’s surface in visible and near infrared in 15 bands [23]. The primary mission of MERIS was to monitor the ocean color to measure chlorophyll concentration for open ocean and coastal areas, and to measure the concentration of yellow suspended matter. In addition, MERIS provided land parameter measurements like vegetation indices and atmospheric parameters. The spatial resolution for land and coast area is 260m by 300m.

The Hyperion instrument is one of three primary instruments on the NASA EO-1 spacecraft. The instrument can image a 7.5 km by 100 km land area per image with 30m spatial resolution, and provide detailed spectral mapping across all 220 channels with high radiometric accuracy.

Airborne Visible Infrared Imaging Spectrometer (AVIRIS) contains 224 different detectors, with 10 nanometer (nm) spectral bandwidth. It delivers 224 spectral channels with the wavelength from 400 nm to 2500 nm, delivering an image cube covering the whole VIS-NIR-SWIR spectrum. The AVIRIS image typically has 4 meter or 20 meter spatial resolution, depending on the altitude at which the instrument was flown. Every time AVIRIS flies, the instrument is used to take several "runs" of data (also known as "flight lines"), in order to cover a substantial area. AVIRIS offers a high signal to noise ratio (SNR) at 500:1, whereas Hyperion offers about 50:1.

AVIRIS and Hyperion have a nominal spectral resolution of 10 nm. The hyperspectral sensing systems record reflected energy from materials at the Earth’s surface, creating a spectrum for every spatial point in the image. These images are functions valued in

\[(x, y, \lambda), \text{ where } x \text{ and } y \text{ represent the two usual image coordinates, and } \lambda \text{ represents the spectral coordinate: hence each image represents a three-dimensional } (x, y, \lambda) \text{ hyperspectral data cube, which lends interesting challenges and opportunities to processing and analysis. Hyperspectral cubes are generated from airborne sensors like the NASA's Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) or the Hyperion sensor on} \]
NASA’s EO-1 satellite. Since the relationship that the magnitude of reflected energy has with the wavelength is also very sensitive to the nature of the target material, a hyperspectral sensor has the advantage that it can distinguish between very subtly related biological targets: it may be used to produce very accurate land cover maps. Moreover, Hyperspectral sensors may be used to distinguish among the major forest species [24, 25], to develop maps of forest chemistry, and to study forestry parameters including vegetative indices. As an example of mapping land-cover types with similar spectral signatures, Goodenough et al. [26] (2003) successfully mapped major forest species with Hyperion data. Although the spectral signatures of these species are very similar, the Douglas-fir (Pseudostuga menziesii), Western Red Cedar (Thuja plicata), Lodgepole Pine (Pinus contorta), and Red Alder (Alnus rubra) species were mapped with 90% average classification accuracy. Hyperspectral data is also used for monitoring forest health by measuring the chemical properties of vegetation such as forest chlorophyll, nitrogen, and leaf water content [27].

1.3.3 Radar Remote Sensing

For completeness we discuss radar, an important sensing method. While it is possible to parallelize radar algorithms, we did not do this in this thesis. Radar is a microwave electromagnetic ranging or distance measuring device, for example: a radar altimeter sends out pulses of microwave signals and records the signal scattered back from the earth’s surface. The microwave region of the electromagnetic spectrum is large and there are several microwave wavelength ranges or bands commonly used in radar remote sensing research for forestry. C-band (3.75cm to 7.5cm wavelength) is common on many airborne research systems, for example CCRS’ Convair-580 and NASA’s AirSAR, and spaceborne systems including Sentinel-1 and Radarsat 1 and 2. Table 4 shows the antenna type including the microwave band and the type of radar polarization measured by each of the AirSAR, PalSAR,
and Radarsat-2 sensors. L-band (15cm to 30cm wavelength) was used onboard NASA’s SEASAT satellite and the Japanese JERS-1 and -2 satellites. NASA’s airborne P-band system (30cm to 100cm wavelength) has the greatest radar bandwidth used in NASA’s AirSAR system [28]. In synthetic aperture radar (SAR) imaging, an antenna transmits pulses of microwave radiation towards the earth surface, and the microwave energy scattered back to the spacecraft is measured. The SAR forms an image by utilizing the time delay of the backscattered signals and the motion of the imaging platform. Moreover, interferometric radar data is useful to measure the height of the surface. Due to the cloud penetrating property of microwave, SAR is able to acquire "cloud-free" images in all weather. This is especially useful for regions which frequently experience cloud cover or are usually dark, like polar areas. Being an active remote sensing device, radar is capable of night-time operation.

**Table 4. Radar Imaging sensors, bands, and polarizations**

<table>
<thead>
<tr>
<th>Satellite</th>
<th>AIRSAR</th>
<th>Convair-580</th>
<th>ALOPS PALSAR</th>
<th>RADARSAT-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>P, L, and C-band</td>
<td>C-band X-band</td>
<td>L-band</td>
<td>C-band</td>
</tr>
<tr>
<td>Polarization</td>
<td>HH, HV, VV, VH</td>
<td>HH/HV+ VV/VH</td>
<td>HH/HV+ VV/VH</td>
<td>HH, HV VV, VH</td>
</tr>
</tbody>
</table>

The apparent roughness of trees and other vegetation varies with the wavelength scale. Hence, they appear as moderately bright features in the image. The tropical rain forests have a characteristic backscatter coefficient between -6 and -7 dB, which is spatially homogeneous and temporally stable. Clear cuts produce less backscatter than the forest canopy, and forest edges are enhanced by shadow and bright backscatter [29]. The dynamic range of the radar backscatter intensity from forest was found to be maximal at P-band and decreases with increasing frequencies [29]. Of the available radar bands, P-band (GHz frequency, 68cm
wavelength) data is most strongly correlated with, and has the highest sensitivity to forest biomass [29-32]. To use AGB retrieval algorithms, confounding effects to be mitigated include soil moisture variability in the boreal forest [29] and topographic variability in tropical forests [30].

In areas with complex structure (various species and multilayer vegetation cover), a combination of microwave images and optical images would be the best solution for determining forest parameters. According to recent biomass estimation studies with microwave images, interferometry yields the best results [31]. In May 2013, a biomass mission using P-band SAR was selected to be the seventh ESA Earth Explorer mission, which is planned for a launch in 2020. This mission will provide the first opportunity to explore Earth’s surface at the ‘P-band’ radar frequency from space. Its primary scientific objectives are to determine the distribution of above ground biomass in the forests and to measure annual changes over the period of the mission [31]. Due to military concerns P-band data will not be collected over Canada, the USA, or Europe.

Beside AGB estimation, delineating burned areas are also useful for estimating carbon emission. The National Forest Carbon Monitoring, Accounting and Reporting System (NFCMARS) takes the burned area map as one of the inputs (Kurz and Apps, 2006) for estimating national carbon emissions. For Radarsat-2 quad-pol data feasibility is demonstrated the for mapping historical fire scars of approximately nine years of age in different forest environments [33, 34], where optical remote sensing is only able to map up to seven years. High-resolution data from Radarsat-2 may improve forest-type mapping using textural analysis.

1.4 High Performance Computing and Remote Sensing

In the last decade revolutionary advances in sensor and computing technology have occurred. Hyperspectral and synthetic aperture SAR instruments now have increased spectral,
spatial, and temporal resolution. Consequently the data sets collected are rapidly increasing in size and the remote sensing data analysis applications are requiring more computing resources [35-37]. For instance, an optical sensor Airborne Visible Infrared Imaging Spectrometer (AVIRIS) contains 224 different detectors with 10 nm spectral bandwidth. It delivers 224 spectral channels (380 nm to 2500 nm wavelengths) in an image cube covering the VIS-NIR-SWIR spectrum. AVIRIS images have 4m (Twin Otter) or 20m (U-2) spatial resolution, according to the altitude at which the instruments were flown. Every time AVIRIS flies, several "runs" of data (also known as "flight lines") are recorded, to capture a substantial area. A full AVIRIS disk can yield about 76 Gigabytes (GB) of data per day. Distributing this data is a problem because some users have bandwidth limitations; others interested in time-series spanning many years will be affected by storage limitations. Therefore, the development of computationally efficient techniques to share the massive amount of remote sensing data and transform the data into scientific understanding is critical for earth observation, environment modeling, and other related areas. The image shown in Figure 7 is AVIRIS data acquired over the Great Victoria Watershed District (GVWD) area in 2001. It was collected by NASA’s Twin Otter aircraft which flew 4 km above the ground, yielding 4m pixels. The image has 3682 samples, 5874 lines, 179 bands (the noise bands are removed), integer data type, and is about 7.21 GB in integer format. At the right, a forest stand spectral profile is shown. The breaks in the profile are due to the removal of zero value bands (noise bands) and atmospheric bands.
Due to the high data rate, satellite ground data processing requires considerable computing power to process data in real-time. As the cost of today’s Commercial Off-The-Shelf (COTS) PC’s and workstations decreases and the performance of network hardware increases, high performance computing (HPC) infrastructure such as clusters [38, 39], distributed networks [38, 40-43], grids [42, 43], clouds [44] and specialized hardware components [45, 46] have been used to provide important architectural developments to disseminate large volumes of remote sensing data [41, 42] and to accelerate the computation in processing raw images [41] and extracting information from remote sensing data [47]. The first commodity cluster, Beowulf cluster, was built in 1994 at NASA’s Goddard Space Flight Center (GSFC) in response to the need for large amounts of computation for processing LANDSAT images. The initial prototype consisted of 16 100Mhz 486DX4-based PC’s which were connected with two hub-based Ethernet networks tied together with channel bonding software so that the two networks acted like one network running at twice the speed. This demonstration cluster showed that one could utilize commodity hardware to build a very cost effective, moderately fast computing platform. Many remote sensing algorithms have been researched that incorporate commodity computer clusters to execute in parallel image
Since then computational performance has increased significantly.

The use of distributed computing systems has rapidly increased for large-scale data-intensive applications and is increasingly leveraging the heterogeneity of computing resources, such as networked data storage and processing resources [40, 43]. There are several advantages to distributed computing, as follows. Distributed computing typically uses existing resources to provide incremental scalability of hardware components. In other words, well-tuned parallel programs can be easily scaled to larger configurations because additional workstations can be added to a distributed computing system. Also, these systems allow analyzing parallel performance on a node-by-node basis. System of Agents for Forest Observation Research with Advanced Hierarchies (SAFORAH) is a distributed data grid created in March 2002 to store and share large volumes of remote sensing data and share these data between various geographically-dispersed research groups such as the Canadian Forest Service (CFS), UVic and other academic and government partners. SAFORAH data stores are distributed across Canada and the computer nodes are connected via high speed fiber-optic links. The system is based on grid technologies and presents the distributed remote sensing data seamlessly and transparently for researchers and the public [42]. The SAFORAH Service Oriented Computational Grid uses the Globus Toolkit [49] as middleware, providing web services such as the Grid-enabled Open Geospatial Consortium (OGC) compliant Web Map Service (WMS), the Web Coverage Service (WCS), and the Web Processing Service (WPS) [42]. End users can access those web services anywhere in a web browser. A parallel framework on Nonlinear Noise Reduction Algorithms for hyperspectral images has been implemented on the SAFORAH grid: this was designed to use Single Instruction Multiple Data (SIMD) architecture to speed up the hyperspectral nonlinear de-noising algorithm. SIMD refers to a parallel computer with one instruction stream, where
the current instruction is executed simultaneously by multiple processing units, each operating separately on different data elements [47]. The concept of the framework is easily adapted and applied to other remote sensing algorithms. Often an image is chopped up into pieces and each piece is sent to individual processor executing the same set of instructions (SIMD). The results are gathered together to form the final image product.

Cloud computing offers a flexible dynamic IT infrastructure [50] and delivers the computing resources (hardware and software) as a service over a network. The fundamental goal of cloud computing is to leverage scalable computing capabilities of virtual machines, quickly process large amounts of data where it is stored, and deliver finished products to the end-user at a low cost. In a Service Oriented Architecture (SOA) end-users request an IT service (or an integrated collection of such services) at the desired functional, quality and capacity level, and receive it either at the time requested or at a specified later time. Infrastructure-as-a-service (IaaS) provides the most basic IT needs such as servers, networking and storage on a usage-based payment model where users can configure virtual machines using customized images to create a customized computing cluster to fit their applications’ needs. Sobie et al. [51] implemented a cloud computing scheduler which provides IaaS service for accessing virtualized computing resources. The scheduler can create many instances of virtual machines and schedule jobs on them. Software-as-a-service (SaaS) is a web service where software applications on the cloud are accessible using a thin client via a web browser. Cappelaere et al. [44] created a full hyperspectral unmixing chain service within a cloud environment accessible through NASA’s EO-1 Web Coverage Processing Service, where the user can perform hyperspectral image classification algorithms in parallel on the cloud.
1.5 Research Objective and Research Questions

Existing High Performance Computing (HPC) parallel remote sensing processing implementations have shown that cloud computing architectures can benefit the processing of large remote sensing images. Cloud computing also can reduce information technology overhead for the end-user, provide greater flexibility through virtual machines (VMs), reduce total cost of ownership, and provide on demand services. The objective of this project is to investigate the use of cloud computing to parallelize remote sensing algorithms, such as forest AGB estimation algorithm, and non-linear hyperspectral image denoising algorithm. In this project, the following main research questions will be addressed:

1. How can we map forest aboveground biomass (AGB) with remote sensing data of forests?
2. What are the requirements for implementing distributed remote sensing image processing in the cloud programming model?
3. Can processing performance for large remote sensing image data be improved by implementation in the cloud environment?

To utilize cloud computing technology a framework will be designed and a Software as a Service (SaaS) created so that users may run remote sensing algorithms in the cloud environment. Since the cloud resource is presented in terms of a list of virtual machines, it gives the user flexibility to adjust the configuration of the VMs used in running an application. The framework work flow is described in a later section. A customized VM image with specific versions of compilers, libraries, and APIs is to be created for running the remote sensing applications in the cloud environment. At the end, the system performance evaluated. A comparison will be made with the other implementations, such as a sequential implementation.
Chapter 2  Aboveground Biomass Estimation Methods

The AGB measurement is used to meet many informational needs: estimating the quantities of timber, supporting bio-energy initiatives, and estimating forest carbon sequestration. In general, AGB is calculated using allometric equations based on measured Diameter at Breast Height (DBH) and/or Height, or converted of forest stock volume. The most accurate field measurement of AGB is a destructive method of tree biomass measurement. However this method is costly for estimating regional and national forest biomass. In the National Forest Inventory (NFI) merchantable biomass and non-merchantable biomass are estimated based on the measurement of permanent and temporary forest ground-plot and photo-plot locations. The merchantable individual tree or stand tree biomass are estimated by the volume-to-biomass model derived by Canadian Forest Service documented in [52].

2.1 National Forest Inventory

The National Forest Inventory program is administered by the Canadian Forest Service and Natural Resources Canada in cooperation with provincial and territorial governments. The permanent and temporary sample plot locations are established by the provinces and territories as an integral part of their respective inventories or for use in growth and yield models. Plot data were obtained from each jurisdiction via data sharing agreements. The 133,786 total permanent and temporary sample plots are located across Canada, representing 7,815,849 trees. There are 11,306,607 individual tree measurements obtained from all provinces and territories [52]. In general, plots were distributed evenly within the forested areas of each jurisdiction, although nationally the distribution is somewhat uneven, e.g., over 90% of plots are in Quebec and British Columbia [52].
The merchantable tree biomass models are based on sample ground plots data and developed or derived from models relating to other province, ecozone, species, genus, and forest types. The individual tree biomass components include stem wood, stem bark, branches, and foliage [52]. Figure 8 shows a picture of the silhouette of a coniferous tree and the four forest components of the tree biomass. The biomass is calculated based on stand layers, which includes merchantable-sized trees, non-merchantable-sized trees, and sapling-sized trees, as shown in Figure 9. AGB for merchantable trees in the National Forest Inventory represents the biomass of growing stock trees taller than 1.3 m and with any diameter at breast height (DBH). The total merchantable tree biomass is calculated according to the total volume of stem wood, the total volume of stem bark, the total volume of the branches, and the total volume of the foliage.

Figure 8. Individual tree biomass contains four components: stem wood, stem bark, branches and foliage. The total of the biomass is the sum of the total volumes of the four components.
2.2 Calculating Large Area AGB with Remote Sensing Images

2.2.1 Vegetation Index

A vegetation index is an indicator that describes the greenness for each pixel in the image in terms of the relative density and health of vegetation. Of the several existing vegetation indices, one of the most widely used is the Normalized Difference Vegetation Index (NDVI). The NDVI is calculated as in the formula:

\[
NDVI = \frac{(NIR - RED)}{(NIR + RED)}.
\]

(1)

NDVI ranges from +1.0 to -1.0. Areas of barren rock, sand, or snow usually show very low NDVI (for example, 0.1 or less). Sparse vegetation such as shrubs and grasslands or senescing crops may result in moderate NDVI (approximately 0.2 to 0.5). High NDVI values
(approximately 0.6 to 0.9) correspond to dense vegetation such as found in temperate and tropical forests, or like crops at their peak growth stage. NDVI is especially useful for continental- to global-scale vegetation monitoring because it can compensate for changing illumination conditions, surface slope, and viewing angle. NDVI values can be averaged over time to establish "normal" growing conditions in a region for a given time of year. The multi-temporal NDVI data in [53] tracked similar seasonal responses for all crops and were highly correlated across the growing season. Further analysis can then characterize the health of vegetation in that place relative to the norm. When analyzed through time, NDVI can reveal where vegetation is thriving and where it is under stress, as well as changes in vegetation due to human activities such as deforestation, natural disturbances such as wild fires, or changes in plants' phonological stage [54].

Regional or global AGB maps may be obtained using vegetation indices calculated from remote sensing data. Many sensors, such as those carried aboard Landsat, MODIS, MERIS and others, measure red and near-infrared light waves reflected by land surfaces. Generally, healthy vegetation will absorb in the red and blue wavelengths, reflect in the green wavelength, and strongly reflect in the near infrared (NIR) wavelength. Unhealthy or sparse vegetation reflects more visible light and less near-infrared light. Bare soils, on the other hand, reflect moderately in both the red and infrared portion of the electromagnetic spectrum [55]. Lu et al. [56] examined the relationships between Landsat TM spectral responses and AGB in East Amazon and found that Landsat TM band 5 is the variable most strongly correlated with AGB.

2.2.2 ND45

It is preferred that permanent sample plot data from the study area are used to develop the biomass relationship with the DBH and height of the tree, ecozone, site class, stand
density and age class extracted from the plots. However, sometimes that information is not available. The AGB estimate created in the previous study over Hinton, Alberta, using vegetation indices from Landsat 5 TM and 7 imagery to create the biomass model, was documented in [57]. A vegetation index ND45 (Normalized Difference) was used on Landsat TM data over Hinton, Alberta to map biomass and aboveground carbon [57]. ND45 takes as input two Landsat bands TM4 and TM5 as shown in equation 2 [57]:

\[ \text{ND45} = 128 \times \left( \frac{(TM4 - TM5)}{(TM4 + TM5)} \right) + 128. \]  

(2)

Next, timber volume in m$^3$/ha is calculated with equation 3[17] for forest area pixels selected as such according to the classification map.

\[ \text{Timber Volume (m}^3/\text{ha}) = -478.58 + 4.5041 \times \text{ND45} \]  

(3)

Next the timber volume was multiplied by the wood density to calculate biomass. Measuring ND45 is a quick way to regional or globally map AGB, and the computation may be automated using open source geospatial libraries such as the Geospatial Data Abstraction Library (GDAL) [58]. The ND45 algorithm is demonstrated in Figure 10.
The algorithm was implemented in the AFT lab using C++ and the GDAL library for a previous project, and selected in this project to be adapted and modified to run on a cloud environment with Hadoop MapReduce. Chapter 5 addresses the detailed implementation.

### 2.2.3 Remote Sensing Image with Inventory Data

To calculate Canada’s AGB, the EOSD land cover map at 25m x 25m spatial resolution was used with inventory models, generating a national aboveground biomass map in 10km x 10km spatial resolution [13]. First the stand height is classified from look-up tables by forest vegetation land cover type, ecozone, site class, stand density and age class extracted from Canada’s Forest Inventory (CanFI). The provincial and territorial permanent sample plot (PSP) and temporary sample plot (TSP) data was compiled and used to derive volume to biomass functions. The merchantable biomass components estimated from plot volume are
total stem wood, total stem bark, total branches, and total foliage. The heights extracted from CanFI were used in the volume to biomass model to estimate the total volume and biomass by ecozone, forest type and density. The AGB mapping was created based on a EOSD 25m x 25m land cover map and then aggregated into a more coarse resolution map. Result biases ranging from 7 t/ha to 57 t/ha from this approach are considered the most reliable biomass mapping of regional and national level with a spatial cell 10,000 ha in size [13].
Chapter 3  Remote Sensing Image Processing and Calibration

Image calibrations or pre-processing operations are performed to correct sensor and platform-specific radiometric and geometric distortions of data before subsequent image analysis. Radiometric correction may be necessary due to variations in scene illumination and viewing geometry, atmospheric conditions, and sensor noise and response. Image calibration may vary depending on the specific sensor and platform used to acquire the data and the conditions during data acquisition. Also, it may be desirable to convert and/or calibrate the data to known (absolute) radiance or reflectance units to facilitate comparisons between data.

3.1 Ground Measurement

Generally for calibration of optical remote sensing images, ground calibration targets such as: farm land, water body, bright and dark targets [27] are measured concurrently with the acquisition of the remotely sensed imagery. In the Evaluation and Validation of EO-1 for Sustainable Development (EVEOSD) project, foliar canopy and ground cover chemistry samples were collected over 54 ground plots within the Greater Victoria Watershed (GVWD) test site. Spectral calibration samples recorded with an ASD field spectrometer represent the following ground targets: the major tree species including Douglas-fir, stacks of foliar samples of ground vegetation (salal) in both illuminated and shady conditions, a field with various grass types, gravel and road targets, an asphalt parking lot, and a dark target [27]. These measurements comprise the EVEOSD spectral library, which has been used to calibrate EO-1 sensor data, and has been used to calibrate other data collected from Landsat-7 and other optical satellites.
3.2 Geometric Correction

Due to the curvature of the Earth and the movement of the sensor platform (e.g., a satellite or airplane) the remote sensing images are distorted with regard to maps. Landsat 4, 5 and 7 are sun-synchronous satellites for which, relative to the image collection path, the Earth rotates from west to east. Geometric rectification or Georectification is the process of removing this image distortion; typically ground control points (GCPs) and appropriate mathematical models are employed. The geometric registration process involves determining the image coordinates of several GCPs clearly identifiable in the distorted image, and matching them to their true map positions in terms of ground coordinates (geographical coordinates). Once several well-placed GCP pairs are identified, the coordinate information is processed by computer to produce a transformation that determines ground coordinates for every pixel in the image. Applying the resulting transformation yields a version of the remotely sensed image expressed in ground coordinates (this process is called “re-projection”). Instead of “re-projecting” an image to express the image in terms of ground coordinates, alternatively we may “re-project” an image expressing its coordinates in terms of those of another image. This is called image-to-image registration, which is often done prior to performing other image transformation procedures, including Georectification.

For large area land cover mapping, pre-processing is often limited to some form of geometric correction.

3.3 Radiometric Correction

Radiometric correction for optical sensor data is intended to mitigate the influence of sun illumination with respect to pixel radiometric response and to permit multisensory integration with calibrated data. Radiometric correction can be completed with calibration by empirical line correction with respect to in-situ ASD measurements [59]. For a national scale
mapping effort a top-of-atmosphere (TOA) approach to account for the influence of sun illumination on pixel radiometric response seems most appropriate [60]. A paucity of atmospheric scattering and absorption data for parameterizing absolute correction procedures (Liang et al. 2002) at the time of image acquisition, and a lack of relative improvements to actual classification performance when more complex approaches were used (Song et al. 2001), were both factors affecting the choice of approach. A TOA-reflectance procedure described in Peddle et al. (2003) based upon Markham and Barker (1986) in combination with image calibration, was the normalization approach adopted for the EOSD land cover project.

3.4 Atmospheric Correction

Many higher-level surface geospatial analyses rely on surface reflectance products such as: vegetation indexes, albedo, Leaf Area Index (LAI), burned area, land cover, and land cover change. The atmosphere is a problem when attempting to measure surface reflectance: the influence of the atmosphere on radiation travelling between the sensor and the ground is very strong and variable. The effect is most pronounced in the visible and near-infrared wavelengths so atmospheric correction is a major issue for visible and near-infrared remote sensing. In order to achieve the reflectance measurement which most accurately represents surface reflectance, atmospheric effects must be accounted for. To this end, FLAASH is a first-principles atmospheric correction software program that makes corrections to spectra at visible light, near-infrared, and shortwave infrared wavelengths.

3.5 Classification Methods

Remote sensing is used widely to shed light on environmental processes by acquiring data representing many kinds of information including: forestry or agricultural land cover information, crop yields and plant health, and in general, information about the dynamics of
vegetation and the forest ecosystem. To acquire reliable forest information, such as land cover information, various types of remote sensing imagery are used to extract useful information by means of classification algorithms. Among the many remote sensing image classification techniques we will consider the three that are the most prominent: K-means clustering, Maximum Likelihood (ML) classification, and the Support Vector Machine (SVM).

### 3.5.1 K-means Clustering

The K-means is a clustering algorithm used widely for unsupervised classification. As a clustering algorithm, K-means partitions the input observations into a number (K) of categories (or clusters) where typically K is a required initial parameter. Implicitly a rule for initializing the cluster means is also required (usually they are initialized randomly). There is deemed to be a representative point (called the cluster mean or centre) for each element of the partition; the K-means algorithm proceeds by optimizing the cluster means. The result of the algorithm is a partitioning of the multidimensional feature space associated with the data observations. For image processing, there are two repeated steps for K-means as follows:

1) Each image pixel is assigned a label corresponding to the nearest cluster mean (according to a distance function). Subsequently 2) each cluster mean is re-computed as: the centroid of all data observations whose labels match the given cluster mean. The revised cluster mean vectors become the inputs for 1), and the two steps continue alternately until no appreciable change of cluster mean vectors is detected between successive iterations of the algorithm. The flow chart is shown in Figure 11 [61].

Most commonly in 1) the Euclidean distance is used as defined in equation (4) below, where \( D_{ij} \) is the distance between the pattern or data vector \( X_i \) with label “i” and a given

\[
D_{ij} = \sqrt{\sum_{k=1}^{n} (x_{ik} - y_{jk})^2}
\]
cluster mean vector $X_j$ with label $j$, $N_b$ is number of image bands, and the data are $N_b$ dimensional vectors $X_i = (X_i^0, X_i^1, \ldots, X_i^{N_b})$ and $X_j = (X_j^0, X_j^1, \ldots, X_j^{N_b})$:

$$D(X_i, X_j) = \sqrt{\sum_{k=1}^{N_b} (X_i^k - X_j^k)}$$  \hspace{1cm} (4)

![Diagram of the K-means Clustering Algorithm]

**Figure 11. The K-means Clustering Algorithm**

K-means clustering was used for unsupervised classification in the EOSD land cover mapping project. The classification procedure shown in Figure 12 involves image pre-processing, the K-means image classification, and finally post-classification evaluation. Each image is pre-stratified into four broad categories (water, non-vegetated, forest conifer, forest deciduous) based on the Normalized Difference Vegetation Index (NDVI). Each of the four categories is then processed with the unsupervised K-means approach. Six of the usual optical Landsat-7 ETM+ bands, as well an additional texture measure, serve as input channels.
for K-means, for a total of 7 input channels used in clustering. The texture measure is an intra-pixel variance derived from the 15m panchromatic band. The classification is initialized with K=241 (241 initial cluster ‘centres’) and the resulting classes are subject to aggregation into 23 labeled classes based on existing ground data and expert feedback. The EOSD classification legend consists of 23 land cover classes (including one “NODATA” class) adapted from the land cover classes developed for the National Forest Inventory (NFI), as shown in Table 2.

![Diagram of classification procedure flow](image)

**Figure 12. EOSD classification procedure flow.**

### 3.5.2 Maximum Likelihood Classification

Maximum Likelihood Classification (MLC), a supervised statistical classifier, is among the most widely used classifiers. The variance and covariance signatures of classes are considered by MLC when assigning pixels to known classes.

Assuming that there are M known classes \( \omega_i \) (where \( i=1,\ldots,M \)) the MLC algorithm is based on the idea of the conditional probabilities \( p(\omega_i|x) \): where \( \omega_i \) is a class (of which there
are M) and x is a measurement vector (which in this case, represents a multispectral or
hyperspectral observation for a given pixel). As if the p(ω_i|x) were known quantities, here is
the classification rule motivating the ML classifier [62]:

\[ x \in \omega_i \text{ if } \left[ p(\omega_i \mid x) > p(\omega_j \mid x) \right] \text{ for all } j \neq i \quad (5) \]

This rule assigns an observation \( x \) to the class \( \omega_i \) for which \( p(\omega_i|x) \) is greatest. Labeled
training data can easily be used to estimate the class conditional probabilities \( p(x \mid \omega_i) \), which
are related to the \( p(\omega_i|x) \) by Bayes theorem:

\[ p(\omega_i|x) = p(x \mid \omega_i) p(\omega_i) / p(x). \quad (6) \]

In (5) some terms remain to be defined: \( p(x) \) is the probability density function (the
probability that the measurement \( x \) exists somewhere in the image) and \( p(\omega_i) \) is the ‘prior’
probability (which represents the unknown fraction of pixels that should be labeled as class
\( \omega_i \)). Combining (5) and (6) gives:

\[ x \in \omega_i \text{ if } \left[ \frac{p(x \mid \omega_i) \cdot p(\omega_i)}{p(x \mid \omega_j) \cdot p(\omega_j)} > \frac{p(x \mid \omega_i) \cdot p(\omega_i)}{p(x)} \right] \text{ for all } j \neq i \quad (7) \]

which we may rewrite as:

\[ x \in \omega_i \text{ if } \left[ g_i(x) > g_j(x) \right] \text{ for all } j \neq i \quad (8) \]

where \( g_i(x) \) is a monotonic function of the \( p(x \mid \omega_i) \cdot p(\omega_i) \), as in (9):

\[ g_i(x) = \ln\{ p(x \mid \omega_i) \cdot p(\omega_i) \} \quad (9) \]

In N-dimensional space the Gaussian distribution is

\[ p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^n |C_i|}} \exp \left(-\frac{1}{2} (x - m_i) C_i^{-1} (x - m_i) \right) \quad (10) \]

where \( m_i \) is the mean vector of data for the class \( \omega_i \):

\[ m_i = E_i \{ x \} \quad (11) \]

and \( C_i \) is the covariance matrix for data in the same class:
When a class is normally distributed, it may effectively be characterized by its mean vector and covariance matrix. The discriminant function \( g_i(x) \) in (9) can be expressed by inserting the exponential form of the normal density function Equation (10), and applying the logarithm:

\[
C_i = E_i \{(x - m_i) (x - m_i)\}
\]

(12)

In (13) we note that a constant term has been removed [62].

Equation (13) is the form used in the ENVI implementation of the classifier[63]. It is used in the following manner: each pixel is selected one at a time from the image, the discriminate functions for every class are computed for that pixel, and the pixel is then allocated to the class for which the discriminate function is maximized (hence we have found the class for which likelihood of membership is maximized).

Note: it is possible to estimate the \( p(x|\omega_i) \) from training data: as the number of times that a pixel with measurement vector \( x \) occurs in the training data, divided by the total number of pixels in the training set. Prior knowledge based on preliminary field tests, maps, historical data and user experience may allow this. When there is insufficient information to make reasonable estimates for the priors, commonly it may be assumed that all the priors represent equal probabilities, as is the case with Equation (14). Classifications based on this assumption tend to favor the most rarely occurring classes.
3.5.3 Sample Size

MLC is a supervised classifier so there should be adequate training samples on the ground that are representative of the desired classes. In a data set with n channels, at least (n+1) training samples should be provided for each class, in order to avoid the possibility of singularity of the sample covariance matrix. In practice it is suggested that, for each class, the training sample size should be at least ten to one hundred times the image dimensionality.

With ground truth for the ML, a species-level forest map may be produced from hyperspectral data. For example, in the EVEOSD project, 54 ground plots around the GVWD test-site were sampled for current (growth during the measurement year) and non-current (growth previous to the measurement year) foliage. On the same site, AISA and AVIRIS hyperspectral images were used to create chlorophyll-a, chlorophyll-b, and nitrogen canopy maps with $R^2$ values of 0.8 and above [59].
A general strategy for supervised classification is depicted in Figure 13 [61]. For Figure 13, the main-processing flow is along the central column; rectangles indicate actions and ellipses indicate results.

### 3.5.4 Support Vector Machine

Support vector machine (SVM) is a supervised classification method for analyzing data and recognizing patterns. The SVM training algorithm takes two classes to build a model which represents the examples as points in space, mapped so that two categories are divided by a clear gap that is as wide as possible. The circled data points that lie closely to the
decision boundary plane \((H)\) as shown in Figure 14 are called “support vectors”, where the marks “+1s” and “-1s” stand for vector observations belonging to the two training data sets.

![Figure 14. The separating hyperplane of classes “-1” and “+1”](image)

A hyperplane \(H\) in a space \(S\) can be written as

\[
\{x \in S | \mathbf{w} \cdot x + b = 0\}, \mathbf{w} \in S, b \in R
\]  

(10)

The dot product \(\mathbf{w} \cdot x\) is defined by

\[
\mathbf{w} \cdot x = \sum_{i=1}^{n} w_i x_i
\]  

(11)

The margin for linear SVM is expressed as the set of pairs \((\mathbf{w}, b)\) that satisfy the following inequalities for all patterns \(x_i\) in the training set[64] [64]:

\[
\begin{cases}
\mathbf{w} \cdot x_i + b > +1 & \text{if } y_i = +1 \\
\mathbf{w} \cdot x_i + b < -1 & \text{if } y_i = -1
\end{cases}
\]  

(12)

Once the margin is determined, new data samples may then be mapped into the same space and predicted to belong to a category, based upon which side of the decision boundary
they fall on. The larger the margin, the better the class separability; hence large margins mean better classifier performance.

In many cases the two classes may not be linearly separable. In this case a transformation function $\varphi$ may be introduced. The left side of Figure 15 shows two classes (labeled as “o” and “x”) that are not linearly separable. The transformation function $\varphi$ is then applied to the original data set, effectively transferring the original data into another higher dimensional feature space. In the higher dimensional feature space, a hyperplane or a set of hyperplanes is created to linearly separate the classes. The transformation function $\varphi$ is known as the “kernel function”.

![Figure 15. SVM kernel function $\varphi$](image)

SVM was developed as a binary classifier, in other words; one application of SVM can only distinguish two classes. For resolution of multiple classes, one approach is to create $N$ two-class applications of SVM, for each of which, the given machine is trained to separate one class from all of the others. Huang examined SVM with two popular kernel functions $\varphi$: polynomial learning machine and radial-basis functions (RBF) applied and tested and compared these incarnations of SVM with other popular classification algorithms including MLC, artificial neural networks (ANN), and decision tree classification (DTC) algorithms, using Landsat TM data for land cover mapping in [65]. Two sets of data inputs were used for this investigation. The red channel, NIR channel, and the NDVI product channel (all derived
from Landsat TM) comprise the input channels used as the first set of data inputs. All six Landsat TM channels plus the NDVI product channel were used for the second set of data inputs. Various quantities of training data (2%, 4%, 6%, 8%, 10%, and 20% of the total image data) were selected to train the algorithms. The overall results show higher accuracy for SVM than for MLC and DTC, but that SVM did not perform significantly better than ANN. The results also showed that increasing the sample size and improving the feature selection can affect accuracy of the classification result. With seven input channels (six TM bands plus the NDVI product) discrimination of land classes is improved [65], relative to using three channels only.

3.6 Accuracy Assessment

In general classification accuracy is estimated using a confusion matrix as shown in Table 5, where each row represents actual observations about a class, and each column represents predictions about a class.

Table 5. Confusion Matrix

<table>
<thead>
<tr>
<th>P(Actual)</th>
<th>P'(Predicted)</th>
<th>N'(Predicted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive</td>
<td>False Negative</td>
<td></td>
</tr>
<tr>
<td>False Positive</td>
<td>True Negative</td>
<td></td>
</tr>
</tbody>
</table>

The overall classification accuracy is the fraction of the total number of test pixels classified correctly, as in Equation (15). The commission error represents false positives. Omission error represents pixels that belong to the truth class but fail to be classified into the proper class (false negatives).
One of the advantages of this performance evaluation tool is it easily reveals the degree to which the model is mislabeling data.

### 3.7 Principal Component Analysis

Classification accuracy is affected by many factors: geometric error, incomplete atmospheric correction, incorrect labeling of clusters after unsupervised classification, incorrect labeling of training sites before supervised classification, indistinguishable classes, training sample size, and the number of input variables. Large sample sizes are usually preferred but are very expensive and also hard to obtain. Since hyperspectral image data is high dimensional, where the adjacent bands are highly correlated, it is a popular practice in remote sensing image processing to apply principal component analysis (PCA) on a high dimensional feature space to achieve dimensionality reduction and produce uncorrelated output bands. A principal component is a linear combination of all the bands of the image. The principal components of the transformed image can be ordered by the eigenvalues associated with the transformation. Typically the bands with higher eigenvalues associated with the PCA transformation are selected as input for subsequent classification processes. The number of principal components is to be determined according to the total percentage of the covariance explained.

### 3.8 Minimum Noise Fraction

The Minimum Noise Fraction (MNF) transformation is an algorithm that transforms normalized linear combinations of the original bands. It produces a linear transformation to
project the data into a new basis, whereby the image signal to noise ratio is maximized. MNF performs two PCA-like steps: the first step performs noise-whitening of the data using the principal components of the estimated “noise covariance matrix”, and then the second step performs the usual PCA upon the “noise whitened” data, after which the dimensionality is examined according to the resulting eigenvalues, leading to possible dimension reduction. The MNF transformation for a given image produces a transformation matrix, which along with the weighted eigenvalue of each band, may be used to calculate the signal to noise ratio (SNR). The bands with high SNR contribute the most information to the image.

### 3.9 Non-linear Noise Reduction

The signal to noise ratio is a concern when working with hyperspectral sensors and high SNR is preferred for analyzing forest canopy chemistry content. Therefore noise reduction is an important preprocessing step for hyperspectral image analysis. Hyperspectral image noise is considered nonlinear. However it is generally treated as linear and there are no easy or efficient methods of nonlinearly denoising an image. Yet, linear noise reduction may be used effectively. Han and Goodenough [34, 66] showed the existence of nonlinearity of noise in hyperspectral images and implemented a non-linear noise reduction program, which removes noise from hyperspectral images and boosts the image SNR. However, the program is very computationally expensive. To improve the processing speed, a parallel framework is created and implemented using a single instruction multiple data stream (SIMD) approach. We will describe the detailed information about the framework in the Chapter 4. By using this non-linear noise reduction, we can improve classification results for hyperspectral-based forest foliage chemical analysis.
Chapter 4  Parallel Hyperspectral Non-linear Noise Reduction Framework

4.1 Introduction

Satellites and sensors collect vast amounts of data for forest monitoring and research. Dealing with such large data sets is a challenge. Therefore many projects adopt data grid techniques. The Canadian Forest Service’s SAFORAH (System of Agents for Forest Observation Research with Advanced Hierarchies) [42] is an example of a project that collects data from a variety of air-borne and space-borne remote sensing sources. SAFORAH was initially created to archive and distribute large geospatial data sets among various research partners at the Canadian Forest Service, at the University of Victoria, and at various other academic and government partner institutions. Recently it has become apparent that providing researchers with the ability to process images on computational grids would significantly enhance SAFORAH. The ability to process existing images to create information products is added to SAFORAH using a service-oriented computational grid based on Globus Toolkit Version 4 (GT4) middleware [49]. The Gavia [67] Project, initiated in 2006 with a CANARIE Intelligent Infrastructure Program (IIP) contract at the University of Victoria (UVic), is a developmental computational grid in a service-oriented architecture. Gavia is based on GT4, incorporating meta-scheduler services based upon Condor. The Gavia Project provides transparent access across the SAFORAH system, with distributed computational facilities in Canada.

Remote sensing image analysis in forest monitoring usually involves large volumes of data, complicated algorithms, and requires a great deal of computational resources. For example, hyperspectral remote sensing images generally involve hundreds of bands. The signal-to-noise ratio (SNR) is a key parameter of satellite sensors; high SNR is always
preferred. Therefore, in hyperspectral image analysis, noise reduction is a vital pre-processing step. The non-linear noise reduction method mentioned in the third chapter is a computationally expensive software algorithm, which requires improvements in terms of computational efficiency.

4.2 Parallel Programming

The study of parallel programming is to determine or identify the parallelism in a task. In other words, operations that can be performed in parallel must be identified. Data dependence graphs are usually used for finding tasks that can be performed independently and can be performed concurrently [68]. Data parallelism refers to situations where the same operations can be applied independently upon distinct elements of a data set. Therefore, a large data set may be split into smaller chunks of data, so that each smaller chunk may simultaneously be assigned for individual processing by one of many processors. Data dependence graphs also exhibit functional parallelism which refers to the situation where different kinds of operations can be applied independently to different distinct data elements.

4.3 Parallel Computing Architectures

Parallel computer architectures depend upon the parallelism with respect to the instruction stream, or with respect to the data stream. Single Instruction Multiple Data stream (SIMD) refers to the parallel computing situation where all of the parallel computing units have the same instructions, yet they execute these instructions upon different data elements. The idea is that many arithmetic units may be at work, all using the same instructions and all performing the same actions. This usually means special multiple data-manipulation instructions are included within the same processing core as the instruction decoder, for the sake of the tight timekeeping.
Multiple Instruction Multiple Data stream (MIMD) parallelism refers to the situation where parallel units execute separate instructions, so that each of them may be doing something different at any given time. One may be adding, another may be multiplying, yet another may be evaluating a branch condition, and so on. This is the type of parallelism obtained with threads, which allow programs to dispatch arbitrary function calls to run on different processors. This paradigm involves multiple fully-featured, independent processing cores, whether they are packaged together on the same chip (multi-core), are packaged separately (multi-processor), or express some kind of hybrid architecture that is a mixture.

In recent years the commodity computing cluster architecture has played a major role in parallel computing. Architectural solutions increase parallelism by building computing clusters with commodity components: tasks become easily parallelizable and scale well with the number of processors involved.

4.4 Parallel Computing for Hyperspectral Images

Hyperspectral images generally have hundreds of bands. Each of these bands is designed to react to the reflected radiation of a specific window of the electromagnetic spectrum; together the bands form the layers of a hyperspectral image. The signal-to-noise ratio (SNR) is the ratio of signal power to noise power, and is a key satellite sensor parameter. SNR quantifies the degree to which the signal has been corrupted by noise. Hyperspectral images with high SNR are preferred by users performing image analysis. Therefore, noise reduction is a vital preprocessing step for hyperspectral image analysis. Software algorithms such as wavelet transforms and linear noise reduction methods may be used to achieve high SNR.

Hyperspectral noise is generally assumed to be linear; however, a hyperspectral image captures a variety of natural phenomena, many of which are non-linear in nature. For example, multiple scattering of solar radiation among targets coupled with variations in sun-
canopy-sensor geometry often are considered the primary sources of the nonlinearity [64]. The presence of a nonlinear attenuating medium such as water is another source of nonlinearity [66]. The heterogeneity of pixel composition contributes to the non-linearity as well [66]. Han and Goodenough [66] showed the existence of nonlinearity in a 4m hyperspectral AVIRIS image by applying statistical analysis methods on selected forest and water pixels (since forest and water are believed to exhibit highly nonlinear characteristics). The experiment found that nonlinearity exists in the spectral domain of the hyperspectral data, but not in the spatial domain [66]. A non-linear noise reduction program was then designed based on Local Geometric Projection, which reduces noise according to the following steps:

1) Construction of state vectors in phase space,
2) Specification of neighborhoods for the state vectors,
3) Elucidation of local projection directions, and
4) Noise reduction by projection of the state vectors upon the local projection vectors.

The locally projective [69] nonlinear noise reduction program “Project” in the TISEAN Time Series Analysis package is used for non-linear image denoising: this program can boost SNR by a factor of two; however, the program is computationally expensive, taking up to four days to finish processing a single image [70]. To make this program useful and utilize nonlinear denoising research a framework is designed to parallelize the process.

This nonlinear parallel processing framework uses the single instruction multiple data stream (SIMD) approach. It allows the use of the legacy code written in other languages, such as FORTRAN. The program is easily parallelized to operate upon multiple image sections, reducing the needs for algorithm rewrites, therefore speeding up the implementation period. The framework is implemented using C++, FORTRAN, the Geospatial Data Abstraction Library (GDAL) libraries version 1.6.1 [58], the TISEAN Nonlinear Time Series Analysis
libraries version 3.0.1 [71], and the Globus Toolkit 4 (GT4). Figure 16 shows the UML class diagram of the non-linear denoising algorithm.

The parallelization framework can be broken down into two parts: a manager and a worker. The manager is responsible for splitting a large image scene into sub-images and is responsible for sending all sub-images to the computing cluster. Worker node(s) individually perform the noise reduction task on a given piece of the image. After an image is de-noised, the manager will merge the processed images back together to form a unified product.

The manager contains four components: the Splitter, the Scatterer, the Gatherer, and the Merger. The detailed descriptions are in the following sections.

The following shows pseudo code of the Manager:

**Manager algorithm**

- Obtain input image
- Obtain number of working node(s)
- Calculate the number of the sub images
- Calculate the number of threads for each node
- Splitter splits the input image into sub-images
  - Scatterer submits the jobs to cluster’s meta scheduler
  - Gatherer collects the results
- If error occurs:
  - Resend the job
- Merger merges resulting images

The workers perform the noise reduction task and stage out the resulting image. This arrangement runs on a cluster; each node performs the denoising process on a sub-image or on several sub-image pieces simultaneously. The de-noising process is executed from a shell script and is summarized according to the following steps:
Worker Algorithm

Check if required libraries are installed
If not, install the libraries
Obtain input sub-image
Run the de-noising program
Generate the de-noised image product
Stage out the de-noised result image
Clean up temporary directory and files

4.4.1 Input Data

The application takes image files in GeoTiff format or ENVI/binary hyperspectral image inputs. There are many data types supported in the GDAL library; the program is able to adapt to several different data types.

4.4.2 Splitter

The Splitter is a program component which runs on the master node and partitions the original image into a specified quantity of sub-images. The Splitter uses GDAL libraries to extract information about the source image and to create the sub-images. The image is partitioned along spatial coordinates, whereas the spectral domain is maintained within each sub-image.
4.4.3 Scatterer

The Scatterer is a program component that runs on the master node and creates a job description, which describes processing details to be carried out on GT 4 based grid nodes. It also creates a GT4 end-point-reference and a shell script for execution on a worker node once it is submitted to the meta-scheduler for each sub-image. The meta-scheduler is responsible for allocating a pool of computational resources and scheduling jobs on the computational grid. The Scatterer is also able to generate job descriptions that are compatible with the Portable Batch System (PBS) scheduler.

4.4.4 Gatherer

The Gatherer is an important component of the program that runs on the master node and it is composed of a Watcher, a Merger and an error handler. It collects de-noised images staged out from the worker nodes, handles job failure and merges the final result. The Gatherer is the most complex portion of the program since it monitors denoised images, handles failures in the de-noising process, and collects final results from each node. After jobs are submitted to the meta-scheduler, the Gatherer is in control of the system. The Gatherer initiates a Watcher process which is a program that is used for checking the stage out of resulting images. The watcher maintains a checklist, containing all sub-image names. When a sub-image result appears, the watcher extracts information from the result set. The result contains a compressed tar file and a dummy file. The dummy file is sent after the tar file and it indicates that the result files has been staged out from the working node. The tar file contains a system log file, an error log file and a processed image file. If the result set is valid, and the error log file is empty and the size of the result image is not smaller than a
predefined value, then the checklist is updated accordingly. If the error log is not empty, or the size of the resulting image is smaller than the predefined value, then a failure occurs, and the job is re-submitted. The maximum number of re-submission attempts may be set according to the contents of a configuration file. After all results are collected, the Gatherer initializes a Merger, which then merges the sub-image results to create a result image product for the entire scene.

The Gatherer has a component “ErrorQueue” used for error handling. This is composed of an “ErrorDetector” and an “ErrorHandler”. The “ErrorDetector” responds to three types of errors: first, after job submission, the Watcher checks if a job is finished by monitoring the directory where the job result will appear. Instead of looking for the result directly, it looks for a dummy file staged out by the worker after staging out the de-noised image. The dummy file is used as an indicator for the Gatherer to indicate completeness for the stage-out process. As soon as the Gatherer sees the dummy file it starts to collect the result: a compressed tar-file. This prevents the Gatherer from collecting the result image in the middle of staging-out. After this the tar-file is decompressed. Three files are expected in the tar archive: an error log, a standard out log, and a de-noised image. The program examines the file size for the error log: if the file size is greater than zero, errors may have occurred during the process. The error handler then flags the sub image for resubmission of the job. If the dummy file does not appear after a given time period, the error handler will consider the task as having failed and will resubmit the sub-task.

4.4.5 Merger

The Merger represents the final activity of gathering and is responsible for mosaicking the sub-images into the final de-noised image product. The de-noised sub-images are checked for correctness and inserted into the final image. The Merger maintains
information about the number of sub-images, as well as file name patterns of the sub-images.

The Gatherer watches for de-noised image results, extracting information from the de-noised image and error log files, and passes the sub-images to the merger for the final assembly. Once the Gatherer has been notified of an accurate returned sub-image file in tar-ball format it calls the Extractor. As implied by its name, the Extractor extracts the sub-image file from the compressed format. The extracted sub-image file is then passed to the Merger, which inserts the data into the final image product, as previously described.

Figure 16. Non-linear Denoising Algorithm: UML Class Diagram

4.4.6 Work Node Process

The program component running on the work nodes consists of a shell script and a C++ based executable. Once active on a particular worker node, the sub-image is written into a text format file using GDAL and passed to the TISEAN Time Series Analysis package for
de-noising. Then the TISEAN package output is written: this represents the resultant de-noised sub-image. Once this is complete, the result is staged back to local storage (along with the dummy-file) and all temporary files are removed.

![Flowchart](image)

Figure 17. Flowchart of the shell script executing on worker node. This controls de-noising program execution.

The shell script is described in Figure 17. The input is a hyperspectral sub-image, which is read, and a destination sub-image of the same size is created. Then the sub-image is read using GDAL by reading all of the bands and columns into one row. Once the row has been translated into a text file, the text file is passed to TISEAN where it is nonlinearly de-noised and output is produced in the same format. The output text file is written into the
destination sub-image in the appropriate row location; the process repeats until all rows of the original sub-image have been processed. The control flow is shown in Figure 18.

![Flowchart](image)

Figure 18. Hyperspectral denoising program: worker nodes flowchart.

4.5 Test Platform

Two test clusters were used for testing: the Mercury cluster located at University of Victoria, and the Venus cluster located at the National Research Council (NRC) in Ottawa. Mercury is a Linux cluster geared towards serial jobs and is composed of 210 IBM blade servers, each with dual-core Intel or AMD processors ranging in speed from 2.4GHz to 3.4GHz. This computing power is fronted by a pool of interactive login nodes running a superset of the software available on the compute nodes, augmented by compilers, libraries, editors, graphical interfaces, and other tools. The Venus cluster has 12 dual-core nodes. Both clusters are grid-enabled and accessible via Gavia and as well as GridWay metascheduler.
4.6 Results

This framework has successfully split, de-noised and created a non-linearly de-noised AVIRIS hyperspectral image product. Figure 19 shows the SNR boost for an AVIRIS image (red colored line). The dotted line represents SNR before non-linear de-noising. The black solid line represents the SNR after de-noising. With the de-noising process, SNR is improved by a factor of two.

![SNR Boost Graph](image.png)

Figure 19. Example of the denoising process for an AVIRIS image

In order to reduce the influence from other cluster users on the grid during testing, the time tests were run on a smaller test dataset which has 100 lines x 100 samples x 179 bands subset of an AVIRIS image, and the results are shown in Figure 20. While these evaluation results show a general decrease in time corresponding to an increase in the number of sub-images, we anticipate the times will be further reduced upon using a dedicated computational grid. As we observed during the testing, the actual times fluctuated depending on the queues of the schedulers.
4.7 Problems and Challenges

The framework shows the benefits of using computational grids for remote sensing algorithms which require intense computational resources. It provides researchers with another resource that can help them to speed up their research activities. The concept of the framework is easy to understand and is easy to apply with other algorithms. It requires less work than rewriting algorithms or designing new algorithms and software libraries, and therefore reduces the cost of developing new software. However, we have encountered some problems with the computational grid when running our de-noising program on the clusters. For example, after a job was submitted to the cluster via the meta-scheduler, the job may end up sitting in the “Pending” state for an indeterminate length of time and need to be cleaned up by the cluster system administrator. Also, because of the heterogeneity of grid computing, the runtime environment may differ between cluster nodes, therefore dynamic installation of required libraries is necessary, which increases the chance of job failure.
Chapter 5 Parallel AGB Generation Using Hadoop Map-Reduce

5.1 Introduction

In the previous grid-enabled parallel implementation exercise we encountered some problems with the computational grid when running our de-noising program on the clusters. First, we had very limited control over the programs that run on the grid node. After the job was submitted to the cluster via the meta-scheduler, the job may stay in the “Pending” state for too long, which means the program has failed. Second, because of the heterogeneity of grid computing, the runtime environment may differ between clusters or between nodes: therefore dynamic installation of required libraries is necessary, increasing the chance of job failure. Third, sometimes the cluster may locate geographically very far away from the job submission location, adding to the network transaction time and bandwidth requirements. Therefore, there is a need to find other ways for parallelizing remote sensing applications.

Hadoop MapReduce [72] is a software framework that emerged recently for cloud computing. It processes vast amounts of data in parallel on large commodity clusters in a reliable, fault-tolerant manner. A MapReduce job usually splits the input data set into independent chunks processed separately by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps which are then input to ‘reduce’ tasks. Typically, both the input and the output of the job are stored in a file-system, such as Hadoop Distributed File System. The framework takes care of scheduling tasks and monitoring them (and re-executing any failed tasks). Although the Hadoop framework is implemented entirely in Java, MapReduce applications need not be.

Hadoop pipes are SWIG-compatible C++ API for implementing MapReduce applications. Having implemented the carbon mapping program with C/C++ in the past, the
Hadoop pipes API was convenient for writing a wrapper program to allow the carbon mapping program to run on the Hadoop MapReduce cluster. In the implementation, a map process is an AGB generation program which takes in a sub-image as the input image, processes it, and generates an AGB map. A reducer process is a program component that takes AGB sub-images and mosaics the resulting images into an AGB map to create a whole scene. The result image of the reducer is then written back to HDFS file system ready for the user to access.

5.2 Hadoop MapReduce Framework

The Hadoop MapReduce framework consists of a single master “ResourceManager”, one slave “NodeManager” per cluster-node, and one “ApplicationMaster” (AM) per application. Applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and/or abstract-classes, job parameters, and job configuration files. The Hadoop job client then submits the job (in jar or executable format) and the job configuration, to the “ResourceManager” which then assumes responsibility for: distributing the software/configuration to the slaves, scheduling tasks and monitoring them, and providing status and diagnostic information to the job-client.

5.3 Job Scheduling

The map tasks are scheduled by “Hadoop YARN”. The fundamental idea of YARN is to split up the functionalities of resource management and job scheduling/monitoring into separate daemons: namely a global “ResourceManager” (RM) and a per-application “ApplicationMaster” (AM). An application is either a single job or a directed acyclic graph (DAG) of jobs. A single Hadoop MapReduce job usually involves one map and one reduce step, where a DAG of jobs may involve multiple map steps and multiple reduce steps. When
a job is submitted to a scheduler, the RM will allocate resources and initiate an “ApplicationMaster” for the job. The AM starts a job container and begins to monitor the jobs. The YARN scheduler is configured via configuration files. The main program crops the full image scene into sub-images, and submits work to the cluster. The AM container monitors the job.

### 5.4 Resource Manager

The “ResourceManager” (RM) and the “NodeManager” (NM) form the data-computation framework. Figure 21 is taken from [73] and modified to show the structure of the RM. The RM is the ultimate authority that arbitrates resources among all the applications in the system. The NM is the per-machine framework agent responsible for containers, monitoring the resource usage, for example CPU, memory, disk, and network usage (these are reported back to the RM).

![Diagram of ResourceManager (RM) and NodeManager (NM)](image)

**Figure 21.** The “ResourceManager” (RM) has two components: the Scheduler and Application Master (AM)
The Scheduler performs its function based on information it collects on memory, CPU, disk, and network usage, and resource requirements of the applications. The Scheduler performs no monitoring or tracking of application status. It also offers no guarantees about restarting failed tasks due to application or hardware failures. The scheduler has a pluggable policy that is responsible for partitioning cluster resources among the various applications and processing queues. Current schedulers such as the capacity Scheduler and the fair Scheduler are some examples of available plug-in schedulers that may be set up via a configuration file. The schedulers ensure each map job is allocated the required resources for the associated map process.

5.5 Application Master

The per-application “ApplicationMaster” (AM) is a Hadoop MapReduce framework specific library. It is responsible to receive job submissions and is responsible for negotiating resources from the “ResourceManager” and must work with NodeManagers to execute and monitor tasks. The AM initiates application containers, monitors jobs, and restarts the AM container in the event of job failure.

5.6 HDFS

The Hadoop Distributed File System (HDFS) is the primary distributed storage infrastructure for Hadoop applications. It is designed to run on commodity computer hardware and is intended for use in applications that need to process large datasets. A HDFS cluster may consist of thousands of computer nodes providing high-throughput access to application data. Each machine in the HDFS system stores portions of the file system: the data is divided into blocks and duplicated across the cluster. HDFS provides high fault-tolerance due to the redundant nature of storage. Usually the HDFS cluster has a NameNode
that manages the file system metadata, whereas there are many DataNodes that are directly responsible for the data storage.

![HDFS Architecture Diagram](image)

**Figure 22. Hadoop Distributed File System (HDFS) Architecture [70].**

The HDFS architecture diagram in Figure 22 shows basic interactions among NameNode, the DataNodes, and clients. Client access of data stored in HDFS involves two steps: firstly the client contacts the NameNode to obtain file metadata and the location of data with respect to the DataNodes; secondly the client performs the actual file I/O directly with the DataNodes. The NameNode stores the HDFS file system information in a file named “fsimage”. However, updates to the file system that add or remove blocks are not reflected in the fsimage file instantaneously. The file system update transactions are recorded to a log file,
from which they are periodically read and applied to the fsimage file. This allows the NameNode to start up faster in the next starting of NameNode.

The Hadoop MapReduce framework provides a list of utilities for HDFS system maintenance operations. For example fsck is used for diagnosing file system health problems or for finding missing files or blocks; Balancer is a tool for re-balancing data distribution among the DataNodes. The Secondarynamenode is responsible for periodically examining the name space at specified checkpoints and helps maintain the log of HDFS file system transactions within certain file size limits at the NameNode.

The HDFS file system is *rack aware*, where a replicated block may be written among data nodes based on the rack information, which includes the topological description of system component connectivity. Master node daemons obtain the rack ID of cluster slaves and take physical positioning of nodes into account while scheduling tasks and allocating storage, see Figure 23.

![Figure 23. HDFS file system checking command. The output shows the file with its locations information.](image)
5.7 Configuring the Cluster

The Hadoop MapReduce clusters are configured using several configuration files, as follows:

- Core-site.xml
- Mapred-site.xml
- Yarn-site.xml
- Hdfs-site.xml

The core-site.xml is a site-specific configuration for a given Hadoop installation. It configures file system such as HDFS, S3, and S3A file systems so that the map reduce program can find out which cluster to be pointing to and where computation is performed. It also sets up the access control and http connections, and monitors the health of the cluster. The hdfs-site.xml file overrides the default HDFS file system setting. Mapred-site.xml contains configuration information that overrides the default values for MapReduce parameters. It controls how map and reduce jobs run on the Hadoop MapReduce framework; for example, how may CPUs and memory will be used for a map and a reduce job. The yarn-site.xml configures the YARN scheduler, ResourceManager (RM) and NodeManager (NM). The “Fair Scheduler” can limit the number of running applications on a per-user basis (or on a per-queue basis) according to entries in the configuration files. This can be useful when a user must make hundreds of simultaneous job submissions, or in general to increase performance when running too many tasks at once would cause difficulties in terms of context-switching, or too much intermediate data being created. Constraining the execution of applications does not cause failure of jobs that are submitted later: when work units are currently busy, applications will wait in the scheduler's queue until computational resources become available. The file yarn-site.xml is edited to customize the “Fair Scheduler” which modifies the respective weights and capacities of existing work queues.
5.8 AGB MapReduce implementation

The MapReduce processing model for the AGB implementation is shown in Figure 24. There are four inputs to the AGB mapping implementation, as follows:

- Landsat image band 1,
- Landsat image band 4,
- Landsat image band 5, and
- the EOSD classification map product, projected into a geographical coordinate system that is consistent to match the above Landsat data products.

The Landsat images are in GeoTiff format, with an individual GeoTiff file for each band. The Landsat images used in this study were downloaded from the U.S. Geological Survey website, and were collected between May 2001 and August 2001.

Figure 24. Data Flow for Hadoop MapReduce Processing Model of ND45
The input GeoTiff files are cropped into a matrix-like grid of small overlapping sub-image tiles, which are systematically given expanded file names that record their row-column coordinates as part of that “matrix”. In order to effectively distribute these tiles according to the MapReduce model, we must determine a correspondence from this data to represent it in the language of the MapReduce framework, which models data exclusively in terms of <key, value> pairs. In terms of how we represented the data, the keys were set to represent successive integer values ranging from 1 up to the total number of “data splits” (which, in the MapReduce framework, is synonymous with the resultant number of map-jobs). For a given key-value pair, the value we determined to be: the comma-separated list summarizing the four sub-image file names (corresponding to the Landsat bands 1,4, and 5, and the EOSD classification map) which represent the given sub-image “tile”. Then, one such <key, value> pair is defined, for each sub-image tile. All the files making up the various sub-images are written to the HDFS system as shown in Figure 24, whereby one set of sub-images (which includes bands 1,4,5 and the EOSD image) for a given “tile” is accessed by the “map” job responsible for processing that “tile”. By passing the list of file names associated with a particular tile, the four sub-image bands received by a map task for processing are unambiguously determined. Moreover, a particular “map” job takes one such image tile (represented by the <key, value> pair received) and produces one or more new <key, value> pairs which are the job outputs. In the MapReduce paradigm <key, value> pairs produced by “map” jobs arrive at one or more “reducers” (in our application, multiple “map” jobs are called to separately process the individual “tiles”; however, we only use one reducer, which is responsible for rearranging the sub-image products into a AGB product for the original scene). In the MapReduce paradigm, each map task emits one or more <key, value> pairs, where the “value” represents data to be passed to the “reducer”, and the “key” represents the identification number for the reducer. Since we only initialize one reducer in
our application, all <key, value> pairs produced by our “map” jobs have the same “key” value: the number “1”. Therefore the sole “reducer” (to which all <key, value> pairs with key “1” are assigned) reads in all of the sub-image tiles, which the “reducer” then rearranges into an AGB map for the original scene.

5.8.1 Map Phase

As mentioned, the MapReduce paradigm operates according to two phases: 1) the map phase, and 2) the reduce phase. For our application the map phase represents performing ND45 calculations and generating AGB maps on a per-tile basis, for each of the constituent image “tiles”. Furthermore, individual “map” jobs (there is one for each image “tile”) are responsible to accumulate a “total carbon” figure for the area under consideration. Upon invocation of a “map” job by the Hadoop MapReduce framework, the “map” job receives and parses the associated <key, value> pair (which, in our application, represents the file names for the input image tiles to be processed, which reside in the HDFS file system). The image files are read from HDFS and “GDALDataset” objects are created via the C++ GDAL library API. The class “GDALDataset”, part of the GDAL library specification, is a presentation of associated image file raster bands that are initialized from image files, and which allows convenient processing and manipulation of image data. The C++ code executed by each of our “map” jobs is responsible to check that the Landsat image bands have the same spatial resolution as the EOSD classification image band. If the resolutions are different, a resampling process is applied to the Landsat images to match the resolution to that of the EOSD classification image. This resampling is performed by the “GDALWarpOperation” class: a high level image-transformation wrapper class within the GDAL library. Next the program calculates the ND45 model for each image pixel (detecting and marking no-data values according to the values in the Landsat bands 4 and 5). A box-car averaging filter is
then applied to the ND45 results, using an 11x11 window size. Then the EOSD classification image is applied as a mask, so that water pixels, other non-forest pixels, and no data values pixels are all filtered out and excluded from the AGB calculations (which are valid only for vegetative areas). Then the AGB map product is then generated for the forest areas. To calculate the mass of carbon associated with the resulting AGB map, the carbon contributions from each pixel at image coordinate \([iX, iY]\) is calculated according to the formula:

\[
(\text{carbon contribution}) = (-478.58 + 4.5041 \times ND45[iX, iY]) \times 409.0 \times 0.5.
\]

The carbon contributions are summed over the area, to yield a total carbon sum for the map area. This total is recorded, and the resulting above ground carbon image product is written back to HDFS file system in GeoTiff format.

**5.8.2 Reduce Phase**

The reduce phase involves the reverse process of the first step (in which the image scene was split into a number of sub-image “tiles”). In the reduce phase (in our application, there is only one “reducer” unit) our reducer is responsible to merge all of the final sub-image above ground carbon data product to generate an above ground carbon data product result for the full original scene. The reducer retrieves all of the carbon sub-image product “tiles” from HDFS and writes the content of each tile into the appropriate sub-image position. Once all the carbon data product “tiles” are written into the appropriate area of the “result” image, the reduce phase has produced a carbon data product for the whole original scene: hence the reduce phase is complete.
5.9 Hadoop Map Reduce Cluster System Configuration

5.9.1 Master/Slave VM Configuration

In this project we configured a Hadoop MapReduce cluster with four compute nodes based on Linux virtual machines (VMs) on a HP Z820 workstation which has forty CPUs, 120GB RAM, and four 4TB Hard drives. Each of our virtual machines is assigned 8 CPU cores, 16 GB RAM, and 150 GB disk space and 2nd 4 Tera byte hard drive, as shown in Table 6. One of the virtual machine instances is configured as the “Master” and the rest of the virtual machine instances are configured as “slaves”. Each VM is equipped with the 64-bit CentOS system version 7 and Hadoop MapReduce version 2.6.2. We administered the MapReduce cluster from a workstation as the Master was using Scientific Linux 6 as the base operating system.

Table 6. Master / Slave VM Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>Number</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>1</td>
<td>8 × 2.5 GHz CPU Node, 16 GB RAM, 150 GB disk space and a 3TB hard drive</td>
</tr>
<tr>
<td>Slave</td>
<td>4</td>
<td>8 × 2.5 GHz CPU Node, 16 GB RAM, 150 GB disk space and a 4TB hard drive</td>
</tr>
</tbody>
</table>

5.9.2 Software Configuration for Cluster Compute Nodes

Each DataNode of the Hadoop cluster is created using a virtual machine image that was created previously. The operating system installed is Centos Linux operating system version 7. Hadoop is version 2.6.2 and is compiled from source code with Oracle JDK
version 1.7.0_67, GCC compiler used is version 4.4.7, and GDAL library was version 1.11.2.

Table 7. Software Configuration for Cluster Compute Nodes (Virtual Machines)

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop</td>
<td>2.6.2</td>
<td>Installed and available for each node (VM instance)</td>
</tr>
<tr>
<td>Centos Linux (core)</td>
<td>7</td>
<td>Pre-configured with Java, GCC, and Hadoop SDKs.</td>
</tr>
<tr>
<td>GCC</td>
<td>4.4.7</td>
<td>For Image Processing Programming</td>
</tr>
<tr>
<td>Java</td>
<td>1.7.0_67</td>
<td></td>
</tr>
<tr>
<td>GDAL lib</td>
<td>1.11.2</td>
<td></td>
</tr>
</tbody>
</table>

5.10 Data

Four scenes consisting of Landsat 7 ETM+ surface reflectance image data during the May to July 2001 interval, were downloaded from the USGS “Earth Explorer” interface. The surface reflectance data are “level-1” products, which means they are considered to be radiometrically calibrated (the radiance values are scaled) and orthorectified using ground control points (for precision correction based on digital elevation models, to correct for relief displacement). Landsat surface reflectance data are generated from the Landsat Ecosystem Disturbance Adaptive Processing System (LEDAPS). This software was originally developed through a National Aeronautics and Space Administration (NASA) “Making Earth System Data Records for Use in Research Environments” (MEaSUREs) grant by NASA Goddard Space Flight Center (GSFC) and the University of Maryland [74]. The scenes considered cover the area of Hinton, Alberta, as shown in Table 8. Figure 25 shows the area covered by the four scenes: the red box indicates the study area of [57] where AGB and above ground carbon computations were conducted over the same Hinton study site area. Due to limited computing resources available at that time, the image area had to be split into 7 small township-based sub-regions. In order to test our re-implementation of the algorithm for the
Hadoop MapReduce framework, we recreate the geographic extent studied in [57] by using four Landsat image scenes. Instead of placing an emphasis upon remote sensing image analysis methods this study focuses on the implementation of generation of AGB by means of the Hadoop MapReduce framework. Landsat bands 4 and 5 were used in the ND45 calculation were preprocessed for removal of clouds and cloud shadows from each dataset. The associated cloud masks and cloud shadow masks were created using cloud maps and cloud shadow maps associated with the data sets.

Table 8. Landsat 7 ETM+ data acquisitions

<table>
<thead>
<tr>
<th>Date</th>
<th>Path/Row</th>
<th>Spatial resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001-07-03</td>
<td>45/23</td>
<td>30m</td>
</tr>
<tr>
<td>2001-07-03</td>
<td>45/22</td>
<td>30m</td>
</tr>
<tr>
<td>2001-05-23</td>
<td>46/22</td>
<td>30m</td>
</tr>
<tr>
<td>2001-07-10</td>
<td>46/23</td>
<td>30m</td>
</tr>
<tr>
<td>2000</td>
<td>EOSD classification</td>
<td>25m</td>
</tr>
</tbody>
</table>

Figure 25. Landsat images and EOSD land cover classification map.

(a): Landsat scenes, over Hinton Alberta (Red box indicates Hinton, Alberta). (b): The EOSD land cover classification map for the Hinton study area.
5.11 The AGB Processing Flow

The main AGB process is automated using a python script which begins by triggering the cropping of the main image into sub-images (of size 4096 lines X 4096 samples) where the sub-images are saved into the HDFS system. The HDFS file system is responsible to copy each sub-image data file into a DataNode (and each data file store is also duplicated across several other “DataNodes”, based on the degree of redundancy that is defined in the configuration file). After the sub-images are saved to HDFS, the python script initiates a Hadoop pipes job, which involves an executable compiled from our C++ program implemented using the C++ Hadoop pipes API. The call to the C++ based program specifies the number of “map” jobs (which we determined to be equal to the number of the sub-images generated) and also specifies the number of “reduce” jobs (which we determined to be one). The C++-based program includes a section that is executed for each “map” job, and a section that is executed for each “reduce” job. Upon invocation of the Hadoop pipes job, the “ResourceManager” negotiates with the “NodeManager” to allocate the resources for the job, together initiating an “Application Manager” (AM). The AM initiates a job container which monitors the map process, restarting the failed map applications in case of job failures.

Figure 26 shows the process flow involved in the code section executed for a “map” job. Each “map” job takes three Landsat bands and an EOSD classification map band, as input. Again, the file names are specified in terms of the <key, value> pair(s) received (one <key, value> pair is received for each unit that executes a “map” job). The “map” program section (as in Figure 24) performs ND45 and carbon map generation on a particular sub-image. Again, image resampling is involved, in case the Landsat 7 ETM+ images have spatial resolution that does not (yet) match that of the EOSD land cover map (the native resolution for the EOSD land cover map is 25m x 25m). The resampling process is implemented with the GDAL library and the cubic convolution (GRA_Cubic) operator belonging to the
“GDALWarpKernel” class. After resampling, the ND45 model is calculated for each pixel of the image using Equation 2 (an ND45 result image is produced). Next the EOSD land cover classification image is applied as a mask for the ND45 image, so that any non-forest pixels are neglected in the subsequent calculation: an AGB image is generated for all forest pixels. The result AGB image is then averaged with an 11 x 11 window. The AGB values for the pixels in the image are multiplied by tree density (using Equation 3) to create an above ground carbon map (for the sub-image area).

![Image of processing flow diagram]

Figure 26. “Map” job: Carbon generation processing flow

### 5.12 Results

Figure 27 image is an aboveground carbon map image generated from the program. Total aboveground carbon in the processed area is 153.6 MT. The right hand side figure of Figure 26 is the legend of the carbon map and indicates the amount of carbon in tons per hectare. The AGB mapping and above ground carbon model calculations were tested using Landsat imagery and processing times are shown in Table 9. The aboveground carbon process used 168 seconds when run sequentially on a HP Z820 workstation which has 40
CPUs, 120GB RAM. The process is also measured on a dedicated Hadoop cluster that used one data node, two data nodes, three data nodes and four data nodes. The result shows the processing time is decreasing when we increase the number of data nodes.

![Legend (T/Ha)](image)

(a) The above ground carbon map  
(b) The legend of carbon map

Figure 27. (a) The above ground carbon map over the map area. (b) The legend of carbon map.

The parallel processing time on one node Hadoop cluster is longer than when it is run sequentially due to the overhead of the Hadoop MapReduce framework. With more nodes the process is faster than when it is run sequentially due to the parallelization. The program will be faster when we process larger images with more nodes.

Table 9. Run time for carbon mapping using Landsat imagery and ND45 model.

<table>
<thead>
<tr>
<th>Landsat Data Samples/lines</th>
<th>Image Size</th>
<th>Time (Seconds) Sequential</th>
<th>Time (Seconds) 1 node</th>
<th>Time (Seconds) 2 nodes</th>
<th>Time (Seconds) 3 nodes</th>
<th>Time (Seconds) 4 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>13271 X 12851</td>
<td>510MB</td>
<td>168</td>
<td>176</td>
<td>153</td>
<td>128</td>
<td>113</td>
</tr>
</tbody>
</table>
Figure 28. Parallelization results showed the increasing performance when using multiple VMs Hadoop MapReduce cluster to process AGB.

5.13 Conclusions

In this chapter the above ground biomass and carbon mapping programs were parallelized based on a Single Instruction and Multiple Data Stream program model in a cloud computing environment. In order to parallelize the AGB program, the Hadoop MapReduce framework was used for job scheduling. To use the Hadoop MapReduce framework, a Hadoop cluster with four VM nodes was configured and the Hadoop Pipes API was utilized to create an interface so that the legacy C++ implementation of a AGB program is runable in the MapReduce framework. A Python script was created to perform splitting images and initializing Hadoop MapReduce jobs. A Hadoop mapper program was written to perform individual AGB mapping, and a reducer program was written to assemble the results into a large image. We have successfully mapped above ground biomass over a large forest area using remote sensing Landsat 7 data. The project showed the possibility of increasing computing efficiency by using more virtual machines to parallelize the AGB process. With two nodes, the processing time is slightly decreased compared to running it sequentially. It
also showed the trend of increasing performance when we increase the number of nodes (Figure 28).

At a more detailed level, some of our accomplishments over the course of the study include:

- Creating a Virtual Machine image featuring the CentOS operating system version 7, the Hadoop Map Reduce Framework 2.6.2, and the GDAL (Geospatial Data Abstraction Library) version 1.11.2.
- Configuration of a Hadoop cluster (using the Virtual Machine image) with four data nodes (one “name node”).
- Implementation of a Hadoop MapReduce wrapper program for mapping above ground biomass using Landsat 7 ETM+ reflectance image, based on C++ using the Hadoop Pipes C++ API interface
- Implementation using the MapReduce model (in terms of identification of map steps and reduce steps) to enable distributed computation of large data via distributed cloud-compute resources.
- Evaluation of the above wrapper implementation, using different numbers of data nodes. [14]

We have demonstrated that virtual machine images represent a simple, comprehensive means for the management of customized programs, libraries, and applications, to fit the user’s special needs. With the Hadoop pipes C++ API we were able to wrap the remote sensing algorithm written in a legacy C/C++ format, and import it into the Hadoop MapReduce framework without substantially rewriting the existing code. We also showed the ability of using the framework for parallelization of the aboveground biomass algorithm for large remote sensing data using the resources available within a cloud-computing environment. The implementation of remote sensing algorithm is simplified with this
framework, which schedules jobs, transmits data across the cloud, and monitors jobs running on the nodes. The Hadoop HDFS file system provides fault tolerance by distributing the data redundantly among the various cloud nodes.

The aboveground biomass and carbon mapping program involves complex computations and hypothetically it could be implemented with multiple iterations of: (i) reading data from HDFS, (ii) applying map and reduce, and (iii) writing back to HDFS. But, this would not be efficient, since communication between map steps and reduce steps is performed by file system operations. Therefore, we used just one level of iteration of the map-reduce paradigm. Implementing multiple layers of iterated Map-Reduce tasks is possible, but the added implementation complexity is not justified because we expect that using multiple layers of iterated Map-Reduce tasks, would introduce unnecessary communication, and decrease performance. With this Hadoop wrapper program implementation, we used only one layer of the MapReduce paradigm, therefore simplifying the implementation of parallel process.

5.14 Future Work

Hadoop MapReduce is a useful technology for processing big data. With this technology framework, we may wrap remote sensing legacy code in a legacy C++ format, to update it to leverage the power of the Hadoop MapReduce application framework running on cloud based distributed computing clusters. Due to the lack of available Hadoop MapReduce cluster resources we only tested the wrapper program on a Hadoop cluster with four nodes (one “NameNode”). In the future, one could expand the tests by adding more nodes to the computing cluster, and increasing the amount of input. One could also use this approach to wrap other remote sensing image processing applications (such as hyperspectral non-linear noise reduction algorithms and radar remote sensing algorithms) in order to expand the
potential methods available to leverage the potential of distributed cloud-computing resources for remote sensing data analysis.

We have observed some performance issues with the Hadoop MapReduce framework including system overheads, and long wait times when the AM container is started. There are also issues with load balancing across the nodes. Furthermore, sometimes the framework gives a “broken pipes” error when running the applications, and system resources are not always fully utilized during job processing (sometimes we observed resource utilization from 25% to 89%). More testing is required to investigate these issues affecting the framework and the wrapper program for larger image analysis.

Newer systems like Apache Spark [75] now improve performance relative to Hadoop by reducing the file system access (intermediate data is not written to the file system). Thus, unnecessarily slow data I/O access bottlenecks between map or reduce are avoided. The Spark streaming model is based on the underlying Spark mechanisms that dynamically match tasks to available resources. This approach yields different kinds of performance characteristics.
Bibliography


Appendix A: Configuration files of Hadoop MapReduce Cluster. The core-site.xml, hdfs-site.xml, mapred-core.xml and yarn-site.xml files are used to save site-specific properties about the Hadoop MapReduce cluster.

1. **core-site.xml**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/mnt/disk/tmp</value>
    <description>Temporary Directory.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://mars1.csc.uvic.ca:54310</value>
  </property>
  <property>
    <name>dfs.datanode.socket.write.timeout</name>
    <value>3000000</value>
  </property>
  <property>
    <name>dfs.socket.timeout</name>
    <value>3000000</value>
  </property>
</configuration>
```

2. **hdfs-site.xml**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/mnt/disk/tmp</value>
    <description>Temporary Directory.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://mars1.csc.uvic.ca:54310</value>
  </property>
  <property>
    <name>dfs.datanode.socket.write.timeout</name>
    <value>3000000</value>
  </property>
  <property>
    <name>dfs.socket.timeout</name>
    <value>3000000</value>
  </property>
</configuration>
```
<configuration>
  
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file://mnt/disk/data/hadoop/hdfs/namenode</value>
    <description>Determines where on the local filesystem the DFS name node should store the name table(fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.</description>
  </property>

  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file://mnt/disk/data/hadoop/hdfs/datanode</value>
    <description>Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices. Directories that do not exist are ignored.</description>
  </property>

  <property>
    <name>dfs.blocksize</name>
    <value>268435456</value>
  </property>

  <property>
    <name>dfs.datanode.socket.write.timeout</name>
    <value>300000</value>
  </property>

  <property>
    <name>dfs.socket.timeout</name>
    <value>300000</value>
  </property>

  <property>

</configuration>
<name>dfs.image.transfer.chunksize</name>
<value>134217728</value>
</property>

<property>
    <name>dfs.namenode.handler.count</name>
    <value>100</value>
</property>

<property>
    <name>dfs.namenode.rpc-bind-host</name>
    <value>0.0.0.0</value>
</property>

<property>
    <name>dfs.datanode.max.xcievers</name>
    <value>4096</value>
</property>

</configuration>

3. **mapred-site.xml**

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Licensed under the Apache License, Version 2.0 (the "License"); you
may not use this file except in compliance with the License. You may obtain
a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless
required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES
OR CONDITIONS OF ANY KIND, either express or implied. See the License for
the specific language governing permissions and limitations under the License.
See accompanying LICENSE file. -->

<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.map.memory.mb</name>
        <value>2028</value>
    </property>
    <property>
        <name>mapreduce.reduce.memory.mb</name>
        <value>6000</value>
    </property>
    <property>
        <name>mapreduce.map.java.opts</name>
        <value>-Xmx2000m</value>
    </property>
</configuration>
```
property>
  <name>mapreduce.reduce.java.opts</name>
  <value>-Xmx3600m</value>
</property>

<property>
  <name>mapreduce.map.cpu.vcores</name>
  <value>2</value>
</property>

<property>
  <name>mapreduce.reduce.cpu.vcores</name>
  <value>3</value>
</property>

<property>
  <name>mapreduce.task.io.sort.mb</name>
  <value>256</value>
</property>

<property>
  <name>mapreduce.task.io.sort.factor</name>
  <value>100</value>
</property>

<property>
  <name>mapreduce.reduce.shuffle.parallelcopies</name>
  <value>5</value>
</property>

<property>
  <name>mapreduce.reduce.shuffle.connect.timeout</name>
  <value>1800000</value>
</property>

<property>
  <name>mapreduce.task.timeout</name>
  <value>6000000</value>
</property>

<property>
  <name>mapreduce.jobtracker.address</name>
  <value>mars1.csc.uvic.ca:54311</value>
  <description>The host and port that the MapReduce job tracker runs at.
  If local and reduce task.</description>
</property>

<property>
  <name>mapreduce.map.log.level</name>
  <value>ALL</value>
  <description></description>
</property>

<property>
  <name>mapreduce.jobhistory.address</name>
</property>
4. yarn-site.xml

<?xml version="1.0"?>
<!-- Licensed under the Apache License, Version 2.0 (the "License"); you
 may not use this file except in compliance with the License. You may obtain
 a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless
 required by applicable law or agreed to in writing, software distributed
 under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES
 OR CONDITIONS OF ANY KIND, either express or implied. See the License for
 the specific language governing permissions and limitations under the License.
 See accompanying LICENSE file. -->

<!-- Site specific YARN configuration properties -->
<property>
    <name>yarn.resourcemanager.address</name>
    <value>mars1.csc.uvic.ca:8032</value>
</property>
<property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>mars1.csc.uvic.ca:8030</value>
</property>
<property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>mars1.csc.uvic.ca:8031</value>
</property>
<property>
   <name>yarn.resourcemanager.admin.address</name>
   <value>mars1.csc.uvic.ca:8033</value>
</property>
<property>
   <name>yarn.resourcemanager.webapp.address</name>
   <value>mars1.csc.uvic.ca:8088</value>
</property>
<property>
   <name>yarn.app.mapreduce.am.job.client.port-range</name>
   <value>50000-51000</value>
</property>
<property>
   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce_shuffle</value>
</property>
<property>
   <name>yarn.resourcemanager.hostname</name>
   <value>mars1.csc.uvic.ca</value>
</property>
<property>
   <name>yarn.nodemanager.resource.memory-mb</name>
   <value>8192</value>
</property>
<property>
   <name>yarn.nodemanager.resource.cpu-vcores</name>
   <value>1</value>
</property>
<property>
   <name>yarn.schedulera.maximum-allocation-vcores</name>
   <value>3</value>
</property>
<property>
   <name>yarn.schedulera.minimum-allocation-vcores</name>
   <value>1</value>
</property>
<property>
   <name>yarn.resourcemanager.scheduler.class</name>
   <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>
</property>
<property>
   <name>yarn.nodemanager.local-dirs</name>
   <value></value>
</property>
<property>
  <name>yarn.nodemanager.log-dirs</name>
  <value>/mnt/disk/logs</value>
</property>

<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>

<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>

<property>
  <name>yarn.app.mapreduce.am.resource.mb</name>
  <value>1228</value>
</property>

<property>
  <name>yarn.app.mapreduce.am.command-opts</name>
  <value>-Xmx983m</value>
</property>

<property>
  <name>yarn.scheduler.minimum-allocation-mb</name>
  <value>256</value>
</property>

<property>
  <name>yarn.scheduler.maximum-allocation-mb</name>
  <value>6000</value>
</property>

<property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>16000</value>
</property>

<property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
  <value>false</value>
</property>

<property>
  <name>yarn.scheduler.fair.user-as-default-queue</name>
  <value>true</value>
</property>

<property>
  <name>yarn.scheduler.fair.preemption</name>
  <value>false</value>
</property>

<property>
  <name>yarn.scheduler.fair.sizebasedweight</name>
  <value>false</value>
</property>
</property>

<property>
    <name>yarn.scheduler.fair.assignmultiple</name>
    <value>false</value>
</property>

</configuration>