Distributed Geo-rectification of Satellite Images using Grid Computing

Y.M. TEO, S.C. Low, S.C. Tay* and J.P. Gozali

Department of Computer Science
*Centre for Remote Imaging, Sensing and Processing
National University of Singapore
Email: teoym@comp.nus.edu.sg
Website: www.comp.nus.edu.sg/~teoym

Outline

- Geo-rectification
- ALiCE Grid Computing System
- Sequential Geo-rectification Algorithm

- Parallelization
  - Task/Data Partitioning
  - Visualization
  - Mapping onto ALiCE

- Preliminary Results
- Summary
Geo-rectification

- correction of skew caused by the earth’s curvature in raw satellite images.

Original Image                                Image after Geo-rectification

ALiCE

- ALiCE (Adaptive and scaLable internet-based Computing Engine)
- Java-based grid middleware - supports development and deployment of grid applications
- Resource discovery and object communication - Java Jini™, JavaSpaces™ or GigaSpaces™
- Maximizes throughput via Job-parallelism
- Maximizes performance via object-parallelism
- Grid programming - object-based template programming
- Multi-language support: Java, C, C++
- ......

ALiCE Layered Architecture

- **Resource Broker**
  - **Object Network Transport Architecture**
  - **Java Technologies: Java Jini & JavaSpaces**
  - **Physical System: Heterogeneous Networks and Machines**

ALiCE Producer-Consumer Architecture

- **Resource Broker**
  - **Scheduler**
  - **Performance Monitor**
  - **Security Manager**

- **Producer**
  - **Registration**
  - **WorkEngine**
  - **Performance Measurement**

- **Consumer**
  - **Registration**
  - **Task Submission**
  - **Result Retrieval**
Object Network-Transport Architecture (ONTA)

- Object-based communication abstraction layer on top of JavaSpaces
- Minimizes space footprint
- Programming - distributed-shared memory

Sequential Geo-rectification

1. Setup parameters
2. Transform co-ordinates
3. Add gridlines & boundaries
4. Finalise image
5. Visualise image
6. Resample image
7. Select GCPs
Sequential Algorithm

Seven main steps:
1. Set up parameters [Setup]
2. Select ground control points [Select]
3. Transform co-ordinates [T]
4. Resample source image [R]
5. Visualize of image [V]
6. Draw gridlines & boundaries [Draw]
7. Finalize corrected image [Finalize]

| Setup | Select | T | R | V | T | R | V | ... | T | R | V | Draw | Finalize |

Parallelization

• **Tasks**
  - Partitioning & distribution for parallel execution
  - Determination of dependencies and impact on execution

• **Data**
  - Determination of data required to support task execution
  - Independent manipulation of data

• **Visualization [not addressed]**
  - Merging of results to current image
  - Adaptation of results for screen display & saving to storage
Three Main Geo-rectification Phases

Setup Phase

- Setting up of parameters
- Selection of GCPs
- These steps makes computations that will be used in the main body itself.
- Dependency on these steps to execute first
- Sequential in nature
- **NOT** to be parallelized!
Execution Phase

- Coordinate transformation
- Image re-sampling
- Visualization of the image
- Abundance of loops
- Computations on independent data sections
- Accounts for 60-90% of execution time

- **candidate** for parallelization!

---

Sequential versus Grid-based Geo-rectification

<table>
<thead>
<tr>
<th>Producer n</th>
<th>T R V</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Producer 3</td>
<td>T R V</td>
</tr>
<tr>
<td>Producer 2</td>
<td>T R V</td>
</tr>
<tr>
<td>Producer 1</td>
<td>T R V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Consumer</th>
<th>Setup</th>
<th>Select</th>
<th>Draw</th>
<th>Finalize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>Setup</td>
<td>Select</td>
<td>T R V T R V T R V ... T R V</td>
<td>Draw</td>
</tr>
</tbody>
</table>

25 May 2004   IPDPS03, Nice, France   13
### Task/Data Partitioning

#### Number of Chunks

\[
\text{#Chunks} = \frac{\text{Width of output image}}{\text{Chunk Width}} \times \frac{\text{Height of output image}}{\text{Chunk Height}}
\]

### Wrapping Chunks into Tasks

#### Task Information

<table>
<thead>
<tr>
<th>Task Width</th>
<th>Chunk Height</th>
<th>Left Corner</th>
<th>Right Corner</th>
<th>...</th>
</tr>
</thead>
</table>
Task/Data Partitioning

- Producer 1
- Producer 2
- Producer 3
- Producer 4

Visualization

- Merging of results to current image
- Adaptation for screen display & saving to storage
- Display of image on the screen
- Output to image (PPM) file
- Output to scientific data (HDF) file
Visualization

- Actual values are then mapped onto the output image

Mapping onto ALiCE Grid

- Partition the image into smaller chunks
- Send these chunks to the Resource Broker
  - Transform co-ordinates
  - Resample source image
  - Collect generated results
- Visualising the image
- Integrating results back into main image
public class swTask implements Task {
    public Result execute() {
        // .. initialize data structures
        for (int j=0; j<chunkHeight; j++) {
            for (int i=0; i<chunkWidth; i++) {
                /* perform coordinate transformation */
                // image resampling */
                return result;
            }
        }
        return result;
    } // end execute()
    // .. other methods
} // end class

public class swGenerator extends TaskGenerator {
    public void generateTasks() {
        // ... initialize variables
        for (int h=0; h<hHeight; h++) {
            for (int w=0; w<wWidth; w++) {
                swTask _swTask=new swTask(); // create task
                process(_swTask); // send task to Res. Broker
                taskGenerated++;
            }
        } // end generateTask()
    // .. other methods
} // end class

public class swTaskListener extends ResultCollector {
    // .. initialize data structure
    public void run() {
        while (resultsCollected<resultsExpected) {
            swTask result=((swTask)swGUI.collectResult()); // collect results
            vResults.add(result); // store result for visualization
        }
    } // end run()
    // .. other methods
} // end class
### Sequential Execution Time

<table>
<thead>
<tr>
<th>No. of tasks</th>
<th>Task Size</th>
<th>Execution Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2000x2000</td>
<td>654.4</td>
</tr>
<tr>
<td>25</td>
<td>1000x1000</td>
<td>636.1</td>
</tr>
<tr>
<td>60</td>
<td>1000x400</td>
<td>609.1</td>
</tr>
<tr>
<td>132</td>
<td>400x400</td>
<td>608.7</td>
</tr>
<tr>
<td>253</td>
<td>400x200</td>
<td>648.8</td>
</tr>
<tr>
<td>506</td>
<td>200x200</td>
<td>836.9</td>
</tr>
</tbody>
</table>

Test Data file = s2000144031151.l2 (108,316,813 bytes)
Resolution = 0.01
Pentium 3: 866MHz, 256MB RAM

### Preliminary Results

<table>
<thead>
<tr>
<th>Test Data</th>
<th>Task Size</th>
<th>Tasks</th>
<th>Sequential Time, $T_s$ (sec)</th>
<th>Speedup</th>
<th>Total Time, $T_p$ (sec)</th>
<th>Speedup</th>
<th>Efficiency $E_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image file 1</td>
<td>400x400</td>
<td>25</td>
<td>106.4</td>
<td>2</td>
<td>56.1</td>
<td>1.89</td>
<td>94.8</td>
</tr>
<tr>
<td>Resolution 0.025</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>42.2</td>
<td>2.52</td>
<td>63.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>40.2</td>
<td>2.65</td>
<td>44.2</td>
</tr>
<tr>
<td></td>
<td>400x200</td>
<td>45</td>
<td>101.5</td>
<td>2</td>
<td>61.8</td>
<td>1.64</td>
<td>82.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>47.7</td>
<td>2.13</td>
<td>53.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>32.0</td>
<td>3.17</td>
<td>52.9</td>
</tr>
<tr>
<td>Image file 1</td>
<td>1000x1000</td>
<td>25</td>
<td>636.1</td>
<td>2</td>
<td>421.3</td>
<td>1.51</td>
<td>75.5</td>
</tr>
<tr>
<td>Resolution 0.025</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>252.8</td>
<td>2.52</td>
<td>63.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>198.1</td>
<td>3.21</td>
<td>55.5</td>
</tr>
<tr>
<td></td>
<td>1000x400</td>
<td>60</td>
<td>609.1</td>
<td>2</td>
<td>308.4</td>
<td>1.98</td>
<td>98.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>176.0</td>
<td>3.46</td>
<td>86.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>128.6</td>
<td>3.74</td>
<td>78.8</td>
</tr>
</tbody>
</table>

25 May 2004
IPDPS03, Nice, France
Summary

- Developed a grid-based geo-rectification application
- Considers the partitioning of task/data and its mapping on a grid system
- Preliminary results show the feasibility of harnessing untapped computing cycles

Other ALiCE Applications

- Geo-rectification of Satellite Images (CRISP)
- Protein Matching and Alignment (BioInformatics Institute)
- Primer Search in Chromosome Sequences (Nanyang Polytechnic)
- Mandelbrot Set
- Cryptography
  - DES Key Cracker
- Distributed Equation Solver
- Progressive Multiple Sequence Alignments (BioInformatics Institute)
- Distributed Ray Tracing
- N-body Problem
- Grid-Enabled Matlab (MIT)
Acknowledgements
1. Singapore-MIT Alliance Programme
2. Sun Microsystems

ALiCE Grid Computing Project
www.comp.nus.edu.sg/~teoym/alice.htm