Crayons – An Azure Cloud Based Parallel System for GIS Overlay Operations

SUSHIL K. PRASAD

COMPUTER SCIENCE DEPARTMENT
GEORGIA STATE UNIVERSITY
Outline

- Geographic Information System (GIS) Data and Computation
- Computational Problems & Our Solution
- Crayons Architecture
  - Centralized Dynamic Load Balancing
  - Distributed Static Load Balancing
  - Distributed Dynamic Load Balancing
- Performance
- Current & Future Work
Raster Vs. Vector Data in GIS

- **Raster Data – Akin to an Image**
  - Divided into cells
  - Each cell represents dominating feature

- **Vector Data – Collection of objects defined using coordinates**
  - Polygons, Lines, Points
  - Better visualization
Spatial Overlay Operations

- Real world GIS data representation through thematic layers
- Layers are overlaid to collect aggregated information
Applications of Spatial Overlay

- **Emergency Response**
  - Hurricane swath’s map overlaid with city’s map to find out safe rescue shelters

- **City Planning**
  - Road network’s map overlaid with public buildings’ map

- **Habitat Analysis**
  - Effect of population density on wildlife habitat
## Typical File Sizes

- GIS Data grows enormously over time
- Beyond the capability of a sequential system
- Can easily take hours to days for processing

<table>
<thead>
<tr>
<th>Source</th>
<th>Example Type</th>
<th>Description</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>US Census (Census.gov 2011)</td>
<td>Block Centroids</td>
<td>Block centroids for entire US</td>
<td>705 MB</td>
</tr>
<tr>
<td></td>
<td>Block Polygons</td>
<td>2000 Block polygons for the state of Georgia</td>
<td>108 MB</td>
</tr>
<tr>
<td></td>
<td>Blockgroup Polygons</td>
<td>2000 Blockgroup polygons for the state of Georgia</td>
<td>14 MB</td>
</tr>
<tr>
<td>GADoT (GDOT 1916)</td>
<td>Roads</td>
<td>Road centerlines for 5-county Atlanta metro</td>
<td>130 MB</td>
</tr>
<tr>
<td>USGS (USGS 1879)</td>
<td>National Hydrography Data set</td>
<td>Hydrography features for entire US</td>
<td>13.1 GB</td>
</tr>
<tr>
<td></td>
<td>National Landcover Data set</td>
<td>Landcover for entire US</td>
<td>3-28 GB</td>
</tr>
<tr>
<td>JPL (NASA 1936)</td>
<td>Landsat TM</td>
<td>pan-sharpened 15m resolution</td>
<td>4 TB</td>
</tr>
<tr>
<td>Open Topography (Open Topography Facility)</td>
<td>LIDAR</td>
<td>LIDAR point clouds 1-4 pts/sq. ft</td>
<td>0.1-1 TB</td>
</tr>
</tbody>
</table>
Computation Problem & Solution

- GIS Data-Intensive and Irregular Computation
- Desktop Sequential Processing is the state-of-the-art
- Our Solution: First end-to-end cloud system
  - Speedup of 30x, skewed small data 10x
  - Scaling to 100 Azure workers
- Open Problems: File i/o, Multicore/GPU algorithms for overlay and R-tree
Why Cloud & Azure

- On-demand Cloud Computing
- Azure Cloud over others
  - Opportunity to research on emerging platform
    - NSF/Microsoft funds
  - Problems
    - Virtual Machine Configuration, reconfiguration, file upload,
    - fast synchronization and messaging among processors,
    - Lack of MPI/Map-reduce libraries
    - Unstable API, Low fidelity of Azure simulator
  - In-built resilience of storage infrastructure
Windows Azure Platform

- **Computation**
  - Web Role
  - Worker Role

- **Storage**
  - Queue Storage – communication
  - Blob Storage – Large data stores
  - Table Storage – Organized data storage
Clipper Library

- Created by Alan Murta
- Supports Union, Intersection, XOR, and Difference operations
- Usable due to open-architecture of Crayons
  - Library is packed as a dll file
- Not Multithreaded – open research problem
Crayons’ Framework

- Three different flavors
  - Centralized Dynamic Load Balancing
  - Distributed Static Load Balancing
  - Distributed Dynamic Load Balancing
Centralized Dynamic Load Balancing

- **Web role does most of the tasks**
  - Download and Parse Files, Create Intersection Graph, and Partition and create tasks

- **Shared task pool for all workers**

- **Demand-supply imbalance**
Centralized Dynamic Load Balancing

Step 1
- Web role
  - Partition Graph
  - Create Intersection Graph
  - Download and Parse Files
  - Read User Selection
  - Commit Output File

Step 2
- Task Pool Queue
- Blob Container
  - Blob
  - Blob
  - Blob
- Worker n
  - Check for work
  - Process Tasks
  - Store Output

Step 3
- Termination Indicator Queue
  - Get msg count
- Task completion message
  - Append output gml file

Put blob IDs
- Get blob IDs
- Put blobs
- Get input files
- Get blobs
Distributed Static Load Balancing

- Web role is relieved of computation
- Workers create work individually
- No sharing of work
- No demand-supply imbalance
- Skewed load distribution can affect performance
Distributed Static Load Balancing

Step 1
Web role

Read User Selection

put file names

Blob Container

get file names

Blob

Worker n
Worker ...
Worker 2
Worker 1

Download and Parse Files

Create Intersection Graph

Process Tasks

Store Output

Blob

Step 2
Termination Indicator Queue

read message count

append output gml

task completion message

Step 3
Commit Output File

flush blocks to gml file

Input Queue
Distributed Dynamic Load Balancing

- Web role still doesn’t do much
- Workers create work
- Work is shared among all workers
Distributed Dynamic Load Balancing
Input Data Sets

- Smaller input data set
  - File Sizes = 16 MB, 770 MB
  - Polygons: Base Layer = 4332, Overlay Layer = 502,674
  - Highly skewed load distribution
Input Data Sets

- Larger input data set
  - File Sizes = 242 MB, 318 MB
  - Polygons: Base Layer = 101,860  Overlay Layer = 128,682
  - Comparatively uniform load distribution
End-to-end Speedups over Small Dataset

![Graph showing speedups over different numbers of worker role instances]

- **Distributed dynamic load balancing**
- **Distributed static load balancing**
- **Centralized dynamic load balancing**
End-to-end Speedups over Large Dataset

![Graph showing speedups for different load balancing methods over the number of worker role instances.]

- **Distributed static load balancing**
- **Distributed dynamic load balancing**
- **Centralized dynamic load balancing**
Individual process timings

- Execution times for small dataset

- Centralized

- Distributed Dynamic

- Distributed Static
Individual process timings

- Execution times for large dataset

![Centralized](image1.png)

![Distributed Dynamic](image2.png)

![Distributed Static](image3.png)
Average timings across all workers

- Supply demand imbalance starts after 16 processors

Centralized Dynamic Load Balancing
Average timings across all workers

- Task creation takes longer as workers are small virtual machines.

![Graphs showing average timings across all workers]

Distributed Static Load Balancing
Average timings across all workers

- Additional reading and writing (from/to Queue) overheads for large number of workers with small data
- Not enough work to process

Distributed Dynamic Load Balancing
Average timings across all workers

- Larger data set

Distributed Dynamic Load Balancing
Engineering Issues

- **Azure-specific [Clouds 2012]**
  - Table vs. Blob
  - Queues – FIFO Behavior
  - Serialization vs. GML Representation
  - Simulator vs. Cloud Environment

- **Concurrency Control in .Net with Azure**
  - Parallel constructs create lots of threads
  - Throttling the cloud storage

- **Clipper library specific**
  - Limited Supported Operations
  - Polygons with Holes can Cause Trouble
Current & Future Work

- Improved task creation using R-Tree
- Avoiding task storage in blobs
- Check pointing
- Parallel polygon overlay algorithm on multicores
- Shape file vs GML format, topologically distributed pages of GIS files

-> Strongly scalable cloud system for skewed data

- MPI & Hadoop based implementation [IPDPS’12]
- Azure benchmark [IPDPS’12]
- GPU based parallel R tree construction

-> GPU based GIS system

Questions? Contact: sprasad@gsu.edu