Performance Optimization of Component-based Data Intensive Applications

Alan Sussman  
Michael Beynon  
Tahsin Kurc  
Umit Catalyurek  
Joel Saltz  
University of Maryland  
http://www.cs.umd.edu/projects/adr

Outline

- Motivation
- Approach
- Optimization – Group Instances
- Optimization – Transparent Copies
- Ongoing Work

Targeted Applications

Pathology  
Volume Rendering  
Surface Rendering  
Groundwater Modeling  
Satellite Data Analysis

Runtime Environment

Heterogeneous Shared Resources:
- Host level: machine, CPUs, memory, disk storage
- Network connectivity

Many Remote Datasets:
- Inexpensive archival storage clusters (1TB ~ $10k)
- Islands of useful data
- Too large for replication

DataCutter

Indexing Service
- Multilevel hierarchical indexes based on spatial indexing methods – e.g., R-trees
  - Relies on underlying multi-dimensional space
  - User can add new indexing methods

Filtering Service
- Distributed C++ component framework
- Transparent tuning and adaptation for heterogeneity
- Filters implemented as threads – 1 process per host

Versions of both services integrated into SDSC SRB
### Indexing - Subsetting

Datasets are partitioned into segments
- used to index the dataset, unit of retrieval
- Spatial indexes built from bounding boxes of all elements in a segment

Indexing very large datasets
- Multi-level hierarchical indexing scheme
- Summary index files -- for a collection of segments or detailed index files
- Detailed index files -- to index the individual segments

### Filter-Stream Programming (FSP)

**Purpose**: Specialized components for processing data
- based on Active Disks research [Acharya, Uysal, Saltz: ASPLOS'98], dataflow, functional parallelism, message passing.
- filters -- logical unit of computation
  - high level tasks
  - init, process, finalize interface
- streams -- how filters communicate
  - unidirectional buffer pipes
  - uses fixed size buffers (min, good)
- manually specify filter connectivity and filter-level characteristics

### Placement

- The dynamic assignment of filters to particular hosts for execution is placement (or mapping)
- Optimization criteria:
  - Communication
    - leverage filter affinity to dataset
    - minimize communication volume on slower connections
    - co-locate filters with large communication volume
  - Computation
    - expensive computation on faster, less loaded hosts

### FSP: Abstractions

**Filter Group**
- logical collection of filters to use together
- application starts filter group instances

**Unit-of-work cycle**
- "work" is application defined (ex: a query)
- work is appended to running instances
- init(), process(), finalize() called for each work
- process() returns {EndOfWork|EndOfFilter}
- allows for adaptivity

### Optimization - Group Instances

Match # instances to environment (CPU capacity, network)

### Experiment - Application Emulator

**Parameterized dataflow filter**
- consume from all inputs
- compute
- produce to all outputs

**Application emulated**
- process 64 units of work
- single batch
Instances: Vary Number, Application

Setup: UMD Red Linux cluster (2 processor PII 450 nodes)
Point: # instances depends on application and environment

Instances: Vary Number, Batch Size

Setup: (Optimal) is lowest all time
Point: # instances depends on application and environment

Adding Heterogeneity

Optimization - Transparent Copies

Runtime Workload Balancing

Use local information:
- queue size, send time / receiver acks
- Adjust number of transparent copies
- Demand based dataflow (choice of consumer)
- Within a host – perfect shared queue among copies
- Across hosts
  - Round Robin
  - Weighted Round Robin
  - Demand-Driven sliding window (on buffer consumption rate)
  - User-defined
### Experiment – Virtual Microscope

- Clients-server system for interactively visualizing digital slides
- Image Dataset (100MB to 5GB per focal plane)
- Rectangular region queries, multiple data chunk reply
- Hopkins Linux cluster – 4 1-processor, 1 2-processor PIII-800, 2 80GB IDE disks, 100Mbit Ethernet
- Decompress filter is most expensive, so good candidate for replication
- 50 queries at various magnifications, 512x512 pixel output

### Virtual Microscope Results

<table>
<thead>
<tr>
<th>R-D-C-Z-V</th>
<th>Average</th>
<th>400x</th>
<th>200x</th>
<th>100x</th>
<th>50x</th>
</tr>
</thead>
<tbody>
<tr>
<td>h-b-b-h</td>
<td>2.10</td>
<td>0.38</td>
<td>0.73</td>
<td>1.73</td>
<td>6.95</td>
</tr>
<tr>
<td>g-g-g-g</td>
<td>1.49</td>
<td>0.37</td>
<td>0.62</td>
<td>1.27</td>
<td>4.60</td>
</tr>
<tr>
<td>g-g-g-g</td>
<td>1.15</td>
<td>0.39</td>
<td>0.50</td>
<td>0.95</td>
<td>3.41</td>
</tr>
<tr>
<td>g-g-g-g</td>
<td>1.15</td>
<td>0.37</td>
<td>0.49</td>
<td>0.95</td>
<td>3.43</td>
</tr>
<tr>
<td>g-g-g-g</td>
<td>1.17</td>
<td>0.39</td>
<td>0.50</td>
<td>0.96</td>
<td>3.50</td>
</tr>
<tr>
<td>g-g-g-g</td>
<td>1.68</td>
<td>0.45</td>
<td>0.68</td>
<td>1.27</td>
<td>5.34</td>
</tr>
<tr>
<td>g-g-g</td>
<td>1.44</td>
<td>0.33</td>
<td>0.58</td>
<td>1.26</td>
<td>4.46</td>
</tr>
<tr>
<td>g-g-g</td>
<td>1.08</td>
<td>0.33</td>
<td>0.45</td>
<td>0.92</td>
<td>3.24</td>
</tr>
</tbody>
</table>

### Experiment - Isosurface Rendering

- UT Austin ParSSim species transport simulation
- Single time step visualization, read all data
- Setup: UMD Red Linux cluster (2 processor PII 450 nodes)

### Sample Isosurface Visualization

\[ V = 0.35 \]
\[ V = 0.7 \]

### Transparent Copies: Replicate Raster

<table>
<thead>
<tr>
<th>1 node</th>
<th>2 nodes</th>
<th>4 nodes</th>
<th>8 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 copy of Raster</td>
<td>12.18s</td>
<td>7.32s</td>
<td>4.17s</td>
</tr>
<tr>
<td>2 copies of Raster</td>
<td>8.16s</td>
<td>5.70s</td>
<td>3.88s</td>
</tr>
</tbody>
</table>

Setup: SPMD style, partitioned input dataset per node
Point: copies of bottleneck filter enough to balance flow

### Experiment – Resource Heterogeneity

- Isosurface rendering on Red, Blue, Rogue Linux clusters at Maryland
  - Red – 16 2-processor PII-450, 256MB, 18GB SCSI disk
  - Blue – 8 2-processor PIII-550, 1GB, 2-8GB SCSI disk + 1 8-processor PIII-450, 4GB, 2-18GB SCSI disk
  - Rogue – 8 1-processor PIII-650, 128MB, 2-75GB IDE disks
  - Red, Blue connected via Gigabit Ethernet, Rogue via 100Mbit Ethernet
- Two implementations of Raster filter – 2-buffer and active pixels (the one used in previous experiment)
Experimental setup

- Read dataset
- Iso surface extraction
- Shade + rasterize
- Merge + view

<table>
<thead>
<tr>
<th>Active Pixel</th>
<th>2-buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5 GB</td>
<td>38.6 MB</td>
</tr>
<tr>
<td>32.0 MB</td>
<td>11.8 MB</td>
</tr>
<tr>
<td>28.5 MB</td>
<td>9.9 MB</td>
</tr>
</tbody>
</table>

Experiment to follow combines R and E filters, since that showed best performance in experiments not shown.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>R node</th>
<th>E node</th>
<th>Ra-M node</th>
<th>Ra node</th>
<th>R-E node</th>
<th>R-E-Ra-M node</th>
<th>RE-Ra-M node</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE-Ra-M</td>
<td>(0) n/a</td>
<td>2.7</td>
<td>4.8</td>
<td>2.9</td>
<td>n/a</td>
<td>11.3</td>
<td>7.7</td>
</tr>
<tr>
<td></td>
<td>(1) 2.2</td>
<td>2.3</td>
<td>4.4</td>
<td>3.0</td>
<td>0.8</td>
<td>7.9</td>
<td>7.9</td>
</tr>
<tr>
<td></td>
<td>(2) 8.2</td>
<td>5.7</td>
<td>3.8</td>
<td>3.2</td>
<td>8.6</td>
<td>8.9</td>
<td>19.0</td>
</tr>
</tbody>
</table>

Only Red nodes used – each one runs 1 RE, 2 RA, and one runs M.

Experimental setup

- 25GB dataset from UT ParSSim (bigger grid than earlier experiments)
- Hilbert curve declustering onto disks on 2 Blue, 2 Rogue nodes
- Skew moves part of data from Blue to Rogue nodes

Skewed data distribution

- Skewed 25% - Active Pixel rendering
  - Configuration: 1 data node
    - Active Pixel algorithm on 8-processor Blue node + Red data nodes
    - Blue node runs 7 Ra or ERa copies and M, Red nodes each run 1 of each except M
  - Active Pixel: 7.3, 3.0, 3.0, 5.7, 2.6, 3.0, 4.4, 3.0, 3.5, 2.8, 2.9, 3.9
  - Skew: 8.2, 4.0, 4.2, 6.5, 4.1, 4.0, 5.1, 4.1, 4.1, 4.0, 3.7, 4.2

Experimental Implications

- Multiple group instances
  - Higher utilization of under-used resources
  - Reduce application response time
  - but ... requires work co-execution tolerance
- Transparent copies can help
  - Most filter decompositions are unbalanced
  - Heterogeneous CPU capacity / load
  - but ... requires buffer out-of-order tolerance
### Ongoing and Future Work

**Ongoing and Future work**

- Automated placement, instances, transparent copies
- Predictive (cost models)
- Adaptive (work feedback)
- Filter accumulator support (partitioning, replication)
- Java filters
- CCA-compliant filters
- Very large datasets – including HDF5 format
  - Using storage clusters at UMD and OSU, then tested at LLNL