Application Emulators and Interaction with Simulators

Tahsin M. Kurc

Computer Science Department, University of Maryland, College Park, MD
Application Emulators

Exhibits computational and access patterns that resemble patterns observed in the real application
Provides a parameterized model of the application
A simplified version
A suite of programs
Why do we need application emulators?

• Trace from actual run
  – It is obtained for a single instance of application and machine configuration
  – It is static, it cannot reflect dynamic nature of application

• Running full application on simulator
  – It complicates the task of simulator unnecessarily
  – Execution of real application requires real data
  – Scaling real application for large scale machines may not be possible

• Application emulator
  – It is parameterized, not specific to a single instance of application/machine configuration.
  – It is a program, it can model dynamic nature of application.
  – Level of abstraction can be controlled, it simplifies task of simulator
  – It does not require real data, can be scaled for large machines
Data-intensive Scientific Applications Suite

Titan
– Satellite data processing
– peer-to-peer

Pathfinder
– Satellite data processing
– client-server (separate IO and Compute nodes)

Virtual Microscope
– Microscope image database server
– data server (multiple simultaneous queries), peer-to-peer
Titan: Input Data Structure

Satellite Data
- Satellite orbits earth in polar orbit
- Each element (IFOV) is associated with a position (in longitude-latitude) and time of recording

Input data is partitioned into data-blocks
- Unit of I/O and communication is a data-block
- Each block contains same number of input elements
- Spatial extent of each block varies
- More overlapping blocks near poles

Data is distributed across disks for I/O parallelism
- Minimax algorithm (Moon et al. 1996) for declustering
Remotely Sensed Data

NOAA Tiros-N w/ AVHRR sensor

AVHRR Level 1 Data
- As the TIROS-N satellite orbits, the Advanced Very High Resolution Radiometer (AVHRR) sensor scans perpendicular to the satellite’s track.
- At regular intervals along a scan line measurements are gathered to form an instantaneous field of view (IFOV).
- Scan lines are aggregated into Level 1 data sets.

A single file of Global Area Coverage (GAC) data represents:
- ~one full earth orbit.
- ~110 minutes.
- ~40 megabytes.
- ~15,000 scan lines.

One scan line is 409
Spatial Irregularity

VHRR Level 1B NOAA-7 Satellite 16x16 IFOV blocks

Longitude

Latitude
Titan: Output Data Structure

2D image

Partitioned into equal size rectangles among processors

Each processor is responsible for processing of blocks that map onto its region
Titan: Processing Loop

While (not done) do
  Issue reads
  Issue receives
  Poll reads
    if (some reads completed) then
      Map data-block to output data
      if (mapped to other processors)
        Issue sends to those processors
      if (mapped to myself)
        Enqueue for processing
  Poll receives
    if (data-block received)
      Enqueue for processing
  Poll sends
  Process a data-block
end while

not done when there are
  * reads yet to be issued
  * pending reads
  * receives yet to be issued
  * pending receives
  * pending sends
  * blocks yet to be process
**Processing Loop**

* All communication and IO are non-blocking operations
* There are dependencies between operations on a data-block

**Life cycle of a data-block**

1. **READ** → **MESSAGE SEND** → **MESSAGE RECEIVE** → **COMPUTE**
2. **MESSAGE SEND** → **MESSAGE RECEIVE** → **COMPUTE**
3. **MESSAGE RECEIVE** → **COMPUTE**
An Emulator for Titan

Input Data Structure
- I/O, Communication, Computation patterns

Output Data Structure (Work load partitioning)
- Communication, Computation patterns

Processing Loop
- I/O, Communication, Computation patterns
An Emulator for Titan

Description of the machine
- number of processors and disks
- machine description file (for Petasim)

Input Data Structure
- Controlled generation of data-blocks using functions
- Parameterized generation of blocks
  - number of blocks
  - size of a block
- Simple block-cyclic distribution of blocks to disks
An Emulator for Titan

Generation of Input data-blocks
An Emulator for Titan

Output Data Structure
- Represented by a 2D rectangle
- Parameterized 2D processor mesh
  - number of processors in x and y dimensions

Processing Loop
- Retain non-blocking nature of operations
- Retain dependencies between operations on a block
- Parameterization of some operations
  - number of maximum pending reads, receives
  - number of blocks processed per iteration of loop
- Each block is assumed to take the same amount of time
  - computation time of a block can be changed
Comparison of Real Application and Emulator

Titan, 10-day data, total number of operations

Titan, 60-day data, total number of operations
Comparison of Real Application and Emulator

Execution times, 10-day data

Execution times, 60-day data
Interaction with Simulators

- Tightly-coupled Simulation
  - Similar to running on real machine
    - a thread is created for each application emulator process
    - emulator performs calls to simulator API for
      - initiating I/O, communication, and computation operations (events)
      - checking their completion
  - Simulator schedules emulator threads to ensure correct logical order of operations
  - Emulator and simulator interacts for each event (e.g., disk read request)
  - Emulator keeps track of dependencies between operations
Interaction with Simulators

Tightly-coupled simulation is not suitable for simulating large scale machines

Number of emulator threads increases with increasing number of processors
  – Scheduling these threads becomes very costly
Message and I/O tables for outstanding non-blocking operations become very large
  – Need for large memory to store these tables
  – Very costly to manage these tables
Each emulator thread has to keep track of non-blocking operations
  – Needs its local data structures (tables) for these operations
  – Replicates the work of simulator.
Interaction with Simulators

Loosely-coupled Simulation

- Idea: Embed application processing loop into simulator
  - Dependency information of processing loop is embedded in the simulator
- Emulator and simulator interacts in distinct phases called “epochs’
  - Emulator sends a set of events (for a set of blocks) to the simulator
  - Simulator processes these events
  - Simulator asks for another set of events from emulator
- One simulator thread and one emulator thread
Interaction with Simulators
(Modeling Dependencies: Work Flow Graphs)

Titan
- DISK READ
- MESSAGE SEND
- MESSAGE RECEIVE
- COMPUTE
- MESSAGE SEND
- MESSAGE RECEIVE
- COMPUTE
- MESSAGE SEND
- MESSAGE RECEIVE
- COMPUTE

Pathfinder
- I/O nodes
- MESSAGE SEND
- MESSAGE RECEIVE
- COMPUTE
- MESSAGE SEND
- MESSAGE RECEIVE
- COMPUTE
- MESSAGE SEND
- MESSAGE RECEIVE
- COMPUTE

Virtual Microscope
- DISK READ
- COMPUTE
- SEND
Interaction with Simulators

No dependencies between operations
– in sets in different epochs
– on different data-blocks

For each block in a set for each processor, emulator passes to simulator
– disk id
  • indicates a read operation from that disk
– length of the block
  • used to estimate I/O and communication time
– list of consumers
  • indicates communication (sends and receives)
– computation time of the block
Comparison of Simulation Models

Accuracy comparison of Tightly-Couple Simulation (TC-SIM) and Loosely-Coupled Simulation (LC-SIM)

<table>
<thead>
<tr>
<th>Emulator</th>
<th>Data set</th>
<th>IBM SP2 Execution Time</th>
<th>TC-SIM Predicted Time</th>
<th>LC-SIM Predicted Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Titan</td>
<td>9K blocks</td>
<td>113</td>
<td>105 (7%)</td>
<td>100 (12%)</td>
</tr>
<tr>
<td></td>
<td>27K blocks</td>
<td>347</td>
<td>322 (7%)</td>
<td>306 (12%)</td>
</tr>
<tr>
<td>Pathfinder</td>
<td>9K blocks</td>
<td>166</td>
<td>153 (8%)</td>
<td>149 (10%)</td>
</tr>
<tr>
<td></td>
<td>27K blocks</td>
<td>497</td>
<td>467 (6%)</td>
<td>452 (9%)</td>
</tr>
<tr>
<td>Virtual Microscope</td>
<td>5K blocks (200 queries)</td>
<td>127</td>
<td>122 (4%)</td>
<td>119 (6%)</td>
</tr>
<tr>
<td></td>
<td>7.5K blocks (400 queries)</td>
<td>243</td>
<td>236 (3%)</td>
<td>234 (4%)</td>
</tr>
</tbody>
</table>
### Comparison of Simulation Models

Predicted execution time and simulation time for TC-SIM and LC-SIM

All results are in seconds for Maryland IBM SP2

<table>
<thead>
<tr>
<th>Emulator</th>
<th>Dataset</th>
<th>P</th>
<th>TC-SIM Predicted Execution</th>
<th>TC-SIM Simulation Time</th>
<th>LC-SIM Predicted Execution</th>
<th>LC-SIM Simulation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27K blocks</td>
<td>32</td>
<td>211</td>
<td>3426</td>
<td>182</td>
<td>6</td>
</tr>
<tr>
<td>Titan</td>
<td>55K blocks</td>
<td>64</td>
<td>285</td>
<td>13154</td>
<td>217</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>110K blocks</td>
<td>128</td>
<td>604</td>
<td>116224</td>
<td>420</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>55K blocks</td>
<td>32</td>
<td>551</td>
<td>11595</td>
<td>496</td>
<td>22</td>
</tr>
<tr>
<td>Pathfinder</td>
<td>110K blocks</td>
<td>64</td>
<td>718</td>
<td>30446</td>
<td>579</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>220K blocks</td>
<td>128</td>
<td>1020</td>
<td>97992</td>
<td>881</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>500 K blocks</td>
<td>32</td>
<td>135</td>
<td>7155</td>
<td>118</td>
<td>4</td>
</tr>
<tr>
<td>Virtual Microscope</td>
<td>1000K blocks</td>
<td>64</td>
<td>145</td>
<td>14097</td>
<td>126</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2000K blocks</td>
<td>128</td>
<td>158</td>
<td>37534</td>
<td>138</td>
<td>17</td>
</tr>
</tbody>
</table>
Conclusions

Emulators for Data-intensive scientific applications
– Simple and parameterized model of applications
– Enables performance prediction studies on large scale machines

Loosely-coupled simulation
– Enables the simulation of large scale machines