Predicting and Controlling Resource Usage in a Heterogeneous Active Network

Virginie Galtier, Yannick Carlinet and Kevin L. Mills (NIST)
Stephen F. Bush and Amit B. Kulkarni (GE CRD)

National Research Council Review Meeting
February 9, 2001
Overview of Presentation

- Modeling an Application’s CPU-Time Use
- Adapting CPU-Time Models for Use by Mobile Code in Heterogeneous Networks
- Applying Adaptive CPU-Time Models (Experiments in Progress)
  - Control Execution of Mobile Code in Magician Execution Environment
  - Predict CPU Consumption among Network Nodes using GE’s Active Virtual Network Management Prediction (AVNMP) System
- Future Research and Related Publications
Modeling CPU Use by Applications

(1) Monitor at System Calls in Node Operating System

(2) Generate Execution Trace

*Trace is a series of system calls and transitions stamped with CPU time use*

- `begin`, `user (4 cc)`, `read (20 cc)`, `user (18 cc)`, `write (56 cc)`, `user (5 cc)`, `end`
- `begin`, `user (2 cc)`, `read (21 cc)`, `user (18 cc)`, `kill (6 cc)`, `user (8 cc)`, `end`
- `begin`, `user (2 cc)`, `read (15 cc)`, `user (8 cc)`, `kill (5 cc)`, `user (9 cc)`, `end`
- `begin`, `user (5 cc)`, `read (20 cc)`, `user (18 cc)`, `write (53 cc)`, `user (5 cc)`, `end`
- `begin`, `user (2 cc)`, `read (18 cc)`, `user (17 cc)`, `kill (20 cc)`, `user (8 cc)`, `end`

(3) Consume Trace & Generate Application Model

**Scenario A:**
- sequence = “read-write”, probability = 2/5

**Scenario B:**
- sequence = “read-kill”, probability = 3/5

**Distributions of CPU time in system calls:**

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>0.8</td>
</tr>
<tr>
<td>write</td>
<td>0.2</td>
</tr>
<tr>
<td>kill</td>
<td></td>
</tr>
</tbody>
</table>

**Distributions of CPU time between system calls:**

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>read-kill</td>
<td>0.67</td>
</tr>
<tr>
<td>begin-read</td>
<td></td>
</tr>
<tr>
<td>read-write</td>
<td>0.33</td>
</tr>
<tr>
<td>kill-end</td>
<td></td>
</tr>
<tr>
<td>write-end</td>
<td></td>
</tr>
</tbody>
</table>
Adapting CPU-Time Models for Mobile Code in Heterogeneous Networks

Each Node Constructs a Node Model using two calibration benchmarks:
- a system benchmark program for each system call, average system time
- for VM, an app. benchmark program average time spent in the VM between system calls

Scaling From Node X to Node Y*

App. model on node X:
read 30 cc
user 10 cc
write 20 cc

Model of node X:
read 40 cc
write 18 cc
user 13 cc

Model of node Y:
read 20 cc
write 45 cc
user 9 cc

App. model on node Y:
read 30*20/40 = 15 cc
user 10*9/13 = 7 cc
write 20*45/18 = 50 cc

* To scale an App. Model in a network, select one Node Model as a reference known by all other nodes

Some Sample Results: Scaling Magician Application Models between Selected Pairs of Nodes vs. Scaling with Processor Speeds Alone

<table>
<thead>
<tr>
<th>AA</th>
<th>Node X</th>
<th>Node Y</th>
<th>Scaling with Models</th>
<th>Scaling with Speeds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ping</td>
<td>Blue Black</td>
<td>&lt;1</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>Blue Green</td>
<td>2</td>
<td>18</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>Blue</td>
<td>Black</td>
<td>&lt;1</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>Black Blue</td>
<td>6</td>
<td>10</td>
<td>92</td>
<td>82</td>
</tr>
<tr>
<td>Red Green</td>
<td>4</td>
<td>14</td>
<td>154</td>
<td>135</td>
</tr>
<tr>
<td>Red Black</td>
<td>6</td>
<td>16</td>
<td>190</td>
<td>163</td>
</tr>
<tr>
<td>Yellow Black</td>
<td>8</td>
<td>15</td>
<td>119</td>
<td>103</td>
</tr>
<tr>
<td>Yellow Green</td>
<td>4</td>
<td>23</td>
<td>24</td>
<td>22</td>
</tr>
<tr>
<td>Black Green</td>
<td>4</td>
<td>23</td>
<td>24</td>
<td>22</td>
</tr>
<tr>
<td>Route</td>
<td>Blue Black</td>
<td>2</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>Black Blue</td>
<td>&lt;1</td>
<td>23</td>
<td>13</td>
<td>32</td>
</tr>
<tr>
<td>Red Green</td>
<td>4</td>
<td>15</td>
<td>88</td>
<td>64</td>
</tr>
<tr>
<td>Red Black</td>
<td>6</td>
<td>19</td>
<td>155</td>
<td>137</td>
</tr>
<tr>
<td>Yellow Black</td>
<td>5</td>
<td>16</td>
<td>190</td>
<td>164</td>
</tr>
<tr>
<td>Yellow Green</td>
<td>6</td>
<td>14</td>
<td>114</td>
<td>83</td>
</tr>
<tr>
<td>Black Green</td>
<td>3</td>
<td>28</td>
<td>26</td>
<td>28</td>
</tr>
<tr>
<td>Blue Green</td>
<td>&lt;1</td>
<td>28</td>
<td>15</td>
<td>204</td>
</tr>
</tbody>
</table>

The Average Absolute Deviation (in Percent) of Simulated Predictions from Measured Reality for Each of Two Active Applications (Average High Percentile Considers Combined Comparison of 80th, 85th 90th, 95th, and 99th Percentiles)
**Application: Control Execution of Mobile Code**

**Experiment in Progress:** Control CPU Usage by Mobile Programs

When mobile code CPU usage controlled with fixed allocation or TTL, malicious or “buggy” mobile programs can “steal” substantial CPU cycles, especially on fast nodes.

When mobile code CPU usage controlled with fixed allocation or TTL, correctly coded mobile programs can be terminated too soon on slow nodes, wasting substantial CPU cycles.
**CPU Control: Expected Results**

### Fast Node

- **X-axis:** Check time points
- **Y-axis:** CPU wasted (ms)
- Data points:
  - TTL
  - NIST

### Slow Node

- **X-axis:** Check time points
- **Y-axis:** CPU wasted (ms)
- Data points:
  - TTL
Can NIST Models enable AVNMP to predict CPU use among heterogeneous network nodes, while providing better look ahead and improved prediction efficiency than simple TTL approaches?
CPU Prediction: Expected Results

**Fast**

- Time check points: 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31
- Rollbacks: 0, 2, 4, 6, 8, 10

**Slow**

- Time check points: 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31
- Rollbacks: 0, 2, 4, 6, 8

**Fast**

- Ahead (ms): 0, 50000, 100000, 150000, 200000, 250000

**Slow**

- Ahead (ms): 0, 50000, 100000, 150000, 200000, 250000
Future Research

- Improve Our Models
  - Model Node-Dependent Conditions
  - Attempt to Characterize Errors Bounds
  - Improve the Space-Time Efficiency of Our Models
  - Continue Search for Low-Complexity Analytically Tractable Models
  - Investigate Models that Continue to Learn

- Investigate Competitive-Prediction Approaches
  - Run Competing Predictors for Each Application
  - Score Predictions from Each Model and Reinforce Good Predictors
  - Use Prediction from Best Scoring Model

- Apply Our Models
  - CPU Resource Allocation Control in Node Operating System
  - Network Path Selection Mechanisms that Consider CPU Requirements
  - CPU Resource Management Algorithms Distributed Across Nodes
Related Publications


