Scalability 1: A Multi-Pronged Approach To Overcoming Scalability Barriers In Paradyn

Philip C. Roth
pcroth@cs.wisc.edu
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706
USA

April 29, 2003
The HPC Situation Today

• Large parallel computing resources
  - Tightly coupled systems
    • Earth Simulator (Japan, 5120 CPUs)
    • HPCx (UK, 1280 CPUs)
    • ASCI Q (LANL, 4096 CPUs)
  - Clusters
    • Pink (LANL, 2048 CPUs)
    • MCR Linux Cluster (LLNL, 2304 CPUs)
    • Aspen Systems (Forecast Systems Lab, 1536 CPUs)
  - Grid
• Large applications
  - ASCI Blue Mountain job sizes (2001)
    • 512 cpus: 17.8%
    • 1024 cpus: 34.9%
    • 2048 cpus: 19.9%
Barriers to Large-Scale Performance Diagnosis

- Computation cost
- Communication cost
- Storage cost

Centralized data collection and processing
Our Approach

1. **MRNet**: infrastructure for building scalable tools
2. **SStart**: strategy for improving tool start-up latency
3. **Distributed Performance Consultant**: strategy for efficiently finding performance bottlenecks in large-scale applications
4. **Multithreaded Data Manager**: for effective front-end performance data management on SMPs
5. **Sub-Graph Folding Algorithm**: algorithm for effectively presenting bottleneck search results for large-scale applications
MRNet: Multicast/Reduction Network

- Software infrastructure for building scalable parallel performance and administration tools
- Scalable data aggregation—reduces centralized data processing cost for tasks like:
  - Computing global performance measures (e.g., CPU utilization for function $F$ across all processes)
  - Collecting application meta-data (e.g., names, addresses of functions in each application process)
- Scalable multicast: efficient delivery of control requests

Details in Dorian’s talk, coming next!
Problem of Tool Start-Up Latency

• Some tools transfer lots of data at tool start-up
  - Debugger needs function names and addresses to set breakpoints by name
  - Paradyn needs information about modules, functions, processes, threads, synchronization objects, call graph

• Front-end bottleneck—high latency:
  - Reduces tool interactivity
  - May cause failures for application’s communication runtime library (e.g., MPI)
SStart: Scalable Tool Start-up

- Reduce redundant data transfer
  - Daemons deliver summary to front end using custom MRNet reduction to find equivalence classes
  - Front end asks equivalence class representatives for complete info
  - Representative daemons send full info to front end
  - **Metrics, code resources, call graph**

- Increase efficiency of non-redundant data transfer
  - In-network concatenation of messages
    - More efficient point-to-point transfers
    - Front-end sees single message instead of many
  - **Metric definition broadcast**
    - Machine resources, daemon info, process info

- Clock skew detection
Clock Skew Detection Algorithm

• Phase 1:
  - Repeated broadcast/reduce pairs to compute each process’ clock skew with directly connected children

• Phase 2:
  - Upward sweep to compute cumulative clock skew to all reachable daemons
Clock Skew Detection: Phase 1

Front-End

Daemon_0

Daemon_1

Daemon_2

Daemon_3
Clock Skew Detection: Phase 1

Front-End
-0.5  3.5

Daemon_0  1  0.5

Daemon_1

Daemon_2

Daemon_3

3.5  0
Clock Skew Detection: Phase 2

Daemon 0

Daemon 1

Daemon 2

Daemon 3
SStart Results
Smg2000 on ASCI Blue Pacific

“Baseline” is the start-up latency without any SStart optimizations

Start-up Latency

- Baseline
- Flat
- 4-way
- 8-way
- 16-way

Start-up Latency vs. Daemons

Time (sec)

0 500 1000 1500 2000 2500

0 100 200 300 400 500 600

Daemons
SStart Results

Smg2000 on ASCI Blue Pacific

SStart Latency

Time (sec)

No MRNet
4-way
8-way
16-way

Daemons
SStart Results

![Graph showing latency results for various activities and machine configurations.](image-url)
### SStart Results

Error of MRNet-based clock skew detection algorithm as compared to direct connection scheme

<table>
<thead>
<tr>
<th>Daemons</th>
<th>Fanout</th>
<th>Average (ms)</th>
<th>Stdev (ms)</th>
<th>By Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Min (ms)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.13325</td>
<td>0.059369079</td>
<td>0.061</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>-0.1755</td>
<td>0.406402202</td>
<td>0.075</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>-0.064125</td>
<td>0.333511221</td>
<td>-0.018</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>-1.47925</td>
<td>0.524946158</td>
<td>-0.486</td>
</tr>
<tr>
<td>64</td>
<td>8</td>
<td>-0.52975</td>
<td>0.844241801</td>
<td>-0.039</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>-4.038492188</td>
<td>0.696410389</td>
<td>0.298</td>
</tr>
<tr>
<td>512</td>
<td>8</td>
<td>-6.511570313</td>
<td>1.167506052</td>
<td>-3.275</td>
</tr>
</tbody>
</table>

(Paradyn's minimum sampling rate is 0.2 s)
Multithreaded Data Manager

- Problem: single-threaded Data Manager in Paradyn front-end is data management bottleneck, limiting overall scalability
- Approach: Data Manager to use multiple threads, taking advantage of increasingly common SMP hardware
  - Part of overall effort to improve performance data management scalability with MRNet, daemon support for Distributed Performance Consultant
- Status: front-end scalability study underway
Problem: front-end bottleneck when searching for performance problems in large-scale applications

- MRNet reduces front-end load when processing *global* performance data (e.g., CPU utilization across all application processes)
- Front-end still processes *local* performance data (e.g., CPU utilization in process 5247 on host blue199.pacific.llnl.gov)
Distributed Performance Consultant

• Approach:

cham.cs.wisc.edu
CPUbound

c001.cs.wisc.edu
myapp367

c002.cs.wisc.edu
myapp4287

c128.cs.wisc.edu
myapp27549

© Philip C. Roth 2003
Distributed Performance Consultant

- Approach:

- CPUbound

- myapp367

- myapp4287

- myapp27549
Distributed Performance Consultant

• Approach:

  Approach: myapp367, myapp4287, myapp27549
Distributed Performance Consultant

• Status: design underway
  - New data management support in daemons
  - New instrumentation cost model
  - New instrumentation scheduling policy
Sub-Graph Folding Algorithm

• Problem: Search History Graph displays grow too complex when showing results from large-scale applications

• Approach: Inspired by the TMC/Sun PRISM parallel debugger, fold similar sub-graphs of the Search History Graph into a single composite sub-graph

• Status: Algorithm designed, feasibility study done, code integration needed

• Dramatic reduction of graph node count in feasibility study – 1232 nodes to 153
Summary

• Taking many approaches to improving Paradyn scalability
  - MRNet
  - SStart
  - Distributed Performance Consultant
  - Multithreaded Data Manager
  - Sub-Graph Folding Algorithm

• Lots yet to do, but efforts paying off
  - SStart reduces start-up latency dramatically
  - MRNet results in Dorian’s talk
Scalability 1: A Multi-Pronged Approach To Overcoming Scalability Barriers In Paradyn

http://www.paradyn.org

paradyn@cs.wisc.edu