Automatic Construction of Coordinated Performance Skeletons
(Predicting Performance in an Unpredictable World)

Jaspal Subhlok  Qiang Xu
University of Houston

NSFNGS 2008
Getting Started

OBJECTIVE: Estimate application performance rapidly in a foreign/dynamic environment, e.g.

- Cluster with upgraded hardware or software components, e.g., MPI Library
- Desktop grid or “Volunteer nodes” or Amazon EC-2 with a shared network
- Execution with different number of processes (8, 16 or more processes best for 8 nodes)
- System under simulation

Motivated by resource selection, mngmt, etc.
Skelton Based Approach?

Build a short running “skeleton” program that mimics execution behavior of a given application

**GOAL:** execution time of a performance skeleton is a fixed fraction of application execution time - *say 1:1000, then...*

If the Application runtime is

<table>
<thead>
<tr>
<th>Application Runtime</th>
<th>Skeleton runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K seconds on a dedicated compute cluster</td>
<td>10 secs</td>
</tr>
<tr>
<td>8K seconds with Open MPI on that cluster</td>
<td>8 secs</td>
</tr>
<tr>
<td>20K seconds on a shared heterogeneous grid</td>
<td>20 secs</td>
</tr>
<tr>
<td>1 million seconds under simulation</td>
<td>1000 secs</td>
</tr>
<tr>
<td>1K seconds on a supercomputer</td>
<td>1 second</td>
</tr>
</tbody>
</table>

*Timed execution of performance skeleton provides an estimate of application performance!*
One Motivation: Mapping Distributed Applications on Networks

Application

Predict performance and select nodes by actual execution of performance skeletons on groups of nodes

Network
How to Construct a Performance Skeleton?

Central challenge in this research

Common sense dictates that an application and its skeleton must be similar in:

- Computation behavior
- Communication behavior
- Memory behavior
- I/O Behavior

All execution behavior is to be captured in a short program
Skeleton Construction

Implementation for parallel MPI codes

**Record Physical Execution Traces**

**Data** → **Mode** → **Sim 2** → **Vis** → **Pre**

**Stream** → **Sim**

**APPLICATION**

**Logicalize Physical Traces into a Single Logical Trace**

**Compress the Logical Trace into Compact Execution Signature**

**Construct Executable Performance Skeleton**
**Logicalization**

**Key challenge:** Identify the dominant communication topology from pairwise node communication matrix

Matching against a known topology is solving graph isomorphism
- No polynomial algorithm

**Practical solution employed:**
1. Match node & edge counts
2. Match eigenvalues
3. Graph Isomorphism algorithm

First two tests eliminate most patterns but cannot prove a match. Exact test used sparingly.
Logicalization Notes

Works well in practice!

- Main communication topology must be static and regular
- Matching only against known patterns, but patterns easy to add and library can be large
  - All n-dim grids or n-ary trees specified in one shot
- Some message exchange not related to main communication pattern observed
  - Ignored with thresholding
  - Can cause inaccuracy, reported to user
- Multiple mixed patterns (equal to subgraph isomorphism) not yet implemented
Goal is to identify loop nests in the trace!

Matching sliding windows of trace is $O(N^3)$. Commonly employed locally on trace sections. So can miss long range repeats (outer loops).

**Two new algorithms developed:**
1. An optimal $O(N^2)$ algorithm (finds outer loops first) : leverages Crochemore’s algorithm to find all repeats
2. Greedy algorithm (finds inner loops first) guaranteed to miss at most 2 iterations of a loop – Very fast
## Loop Discovery Performance

<table>
<thead>
<tr>
<th></th>
<th>Raw Trace Length (MPI Calls)</th>
<th>Compressed Trace Length (MPI Calls)</th>
<th>Optimal Loop Discovery (seconds)</th>
<th>Greedy Loop Discovery (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BT</td>
<td>17106</td>
<td>44</td>
<td>311.18</td>
<td>8.91</td>
</tr>
<tr>
<td>SP</td>
<td>26888</td>
<td>89</td>
<td>747.73</td>
<td>7.61</td>
</tr>
<tr>
<td>LU</td>
<td>323048</td>
<td>63</td>
<td><strong>113890.21 (~30 hours)</strong></td>
<td><strong>61.9</strong></td>
</tr>
<tr>
<td>CG</td>
<td>41954</td>
<td>10</td>
<td>240.27</td>
<td>8.48</td>
</tr>
<tr>
<td>MG</td>
<td>10047</td>
<td>648</td>
<td>144.54</td>
<td>10.88</td>
</tr>
</tbody>
</table>
Validation of Skeleton Construction

Skeletons constructed for Class C NAS MPI benchmarks up to 128 nodes

Skeletons employed to predict performance in a variety of new scenarios

- Execution with different number of nodes for the same number of processes
- Execution under varying available bandwidth
- Execution under competition with other jobs
- Execution on a different clusters
- Execution under a new MPI library (Open MPI)
Validation Results

Skipping the large suite of graphs!

For most applications and scenarios, the prediction was rather accurate with error within 10% for skeletons running for a few minutes.

However:

• Prediction with competing jobs inaccurate!
• Some scenarios showed high errors (> 20%) in particular CG benchmark.

Reasons:

1. Computing not modeled precisely (memory, instructions)
2. Synchronization impact can exaggerate variations
Conclusions

- Performance skeletons are an effective tool for estimating performance dynamically
- Methodologies for logicalization and loop nest discovery have broad applicability

Open to collaborations!
Thanks to NSF

FOR MORE INFORMATION:
- www.cs.uh.edu/~jaspal    jaspal@uh.edu
Prediction Accuracy of Skeletons
(average across all sharing scenarios)

Error (%) vs. Skeleton Duration:
- 10 second skeleton
- 5 second skeleton
- 2 second skeleton
- 1 second skeleton
- 0.5 second skeleton

Legend:
- BT
- CG
- IS
- LU
- MG
- SP
- Average
Prediction for Different Sharing Scenarios
(10 second skeletons)

Error is higher with network contention
- communication is harder to scale down and affects synchronization more directly
Average Prediction: Average slowdown of entire benchmark used to predict execution time for each program.

Class S Prediction: Class S benchmark (~1 sec) programs used as skeletons for Class B (30-900s) benchmarks.

Even the smallest skeletons are far superior!