Prophesy: Automating the Modeling Process

Valerie Taylor, Xingfu Wu, Jonathan Geisler,
Xin Li, Zhiling Lan
Northwestern University

Rick Stevens, Ivan Judson, Mark Hereld
Argonne National Laboratory

SPONSORS: National Science Foundation, National Aeronautics and Space Administration
Problem Statement

Given:

- **Performance models and analyses are critical**
  - Requires significant development time
- **Distributed systems are complex**
- **Some applications require distributed systems**

Goal

- **Efficient execution of distributed applications**

Proposed Solution

- **Automate as much as possible**
- **Community involvement**
Prophesy System

DATA COLLECTION

DATABASES

DATA ANALYSIS

Profiling & Instrument.

Actual Execution

Template Database

Performance Database

Systems Database

Model Builder

Symbolic Predictor
Automated Instrumentation

- In-line data collection
- Instrument at the level of basic loops
- Allow for user-specified instrumentation

$T = E \times f$

```c
for (I=1; I<N; I++){
    V(I) = A(I) \times C(I);
    B(I) = A(2I + 4);
}
```

$T = E \times f$

```c
INSTRUMENTATION CODE
for (I=1; I<N; I++){
    V(I) = A(I) \times C(I);
    B(I) = A(2I + 4);
}
```

INSTRUMENTATION CODE
Automated Instrumentation

Source Code → Parser → Instrumented Source Code → General Compiler → Instrumented Executable

Performance Relations → Call Graph

Performance Files → Prophesy Database
Applications

Application
(Molecular Dynamics)

Module
(md.c)

Function
(check_radius)

Basic Unit
(For(…))

Module
(partition.c)

Function
(comp_inter)

Basic Unit

Module

Basic Unit

Basic Unit
Databases

- Hierarchical organization
- Organized into 4 areas:
  - Application
  - Executable
  - Run
  - Performance Statistics
Prophesy Database

Application

Executable

Run

Application Performance

Modules

Inputs

Function

Function Performance

Functions

Systems

Basic Unit Performance

Resource

Connection

Data Structure Performance

Compliers

Model Template

Model_Info

Control Flow

Library
**User Input**

- **User must first register an application**
- **Requires information about application and modules**
- **Done once for each executable**
Data Analysis

- Develop performance models
- Make predictions
- Performance tune codes
- Identify best implementation
- Identify trends
Automated Modeling Techniques

- **Utilize information in the template and system databases**
- **Currently include three techniques**
  - Curve fitting
  - Parameterization
  - Composition using coupling values
Curve Fitting Method

- Use least squares
- Use database information:
  - Empirical data
  - Computational complexity of dominate function
- Easy to generate, but does not expose system parameters
Curve Fitting: Usage

Analytical Equation
(Matlab: LSF)

Matrix-matrix multiply:
LSF : 3

Performance
Data

Application Performance
Function Performance
Basic Unit Performance
Data Structure Performance

Model Template

APART '01
Curve Fitting: Example

- **Matrix-matrix multiply**
- **SGI (8 processors)**
- **Used data for 100 < rank < 800**

\[
3.88 \times 10^{-8} N^3 - 1.255 \times 10^{-5} N^2 + 0.0028N - 0.1498
\]
Parameterization Method

- **Manual analysis of the kernel or function**
  - Hand count operations
  - Expose system parameters
  - Only done once per kernel

- Implementation-dependent

- Utilize database information:
  - System parameters in system database
  - Model and scripts in template database

- Number of kernels is bounded
Parameterization: Usage

Analytical Equation

\[(2N/(P-1)) \times \text{recv}\]

System Data:

recv

Model Template

Matrix-matrix multiply:

Parameterization:

\[(2N/(P-1)) \times \text{recv}\]
**Parameterization: Example**

- **Matrix-matrix multiply**
- **SGI (8 processors)**

![Graph showing execution time vs matrix rank]

\[ \frac{N^3}{(P-1)} T_{\text{comp}} + \frac{2N}{P-1} T_{\text{op}} + NT_{bc} \]
Kernel Coupling

- Two kernels (i & j)
- Three measurements
  - $P_i$: performance of kernel i isolated
  - $P_j$: performance of kernel j isolated
  - $P_{ij}$: performance of kernels i & j coupled

- Compute $C_{ij} = \frac{P_{ij}}{P_i + P_j}$
**Coupling Categories**

- $C_{ij} = 1$: no coupling
- $C_{ij} > 1$: destructive coupling
- $C_{ij} < 1$: constructive coupling
**Coupling Categories**

\[ C_{ij} = 1: \text{No Coupling} \]

- **Kernel A**
  - Shared Resource
- **Kernel B**
  - Shared Resource

\[ C_{ij} < 1: \text{Constructive Coupling} \]

- **Kernel A**
  - Shared Resource
- **Kernel B**
  - Shared Resource

\[ C_{ij} > 1: \text{Destructive Coupling} \]

- **Kernel A**
  - Shared Resource
- **Kernel B**
  - Shared Resource
Extending Coupling Work

- Use weighted averages to determine how to combine coupling values
- Example:
  - Given you have the coupling values

\[

t = \alpha_1 E_A + \alpha_2 E_B + \alpha_3 E_C
\]

\[
\alpha_1 = \frac{(C_{AB} \cdot P_{AB} + C_{AC} \cdot P_{AC})}{P_{AB} + P_{AC}}
\]
Composition Method: Example

- Synthetic kernels (array updates)

Kernel A (196.44)

Kernel B (207.16)

Kernel C (574.19)

<table>
<thead>
<tr>
<th>Kernel Pair</th>
<th>Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - B</td>
<td>0.97</td>
</tr>
<tr>
<td>B - C</td>
<td>0.75</td>
</tr>
<tr>
<td>C - A</td>
<td>0.76</td>
</tr>
</tbody>
</table>

$\alpha_1 = 0.8472 \quad \alpha_2 = 0.8407 \quad \alpha_3 = 0.7591$

Actual total time: 799.63s

Coupling time: 776.52s (Error: 2.89%)

Adding individual times: 971.81s (Error: 23%)
Composition Method: Usage

Adjacent Kernels

- Model Template
- Control Flow
- Functions

Data and System Info

- Run
- Inputs
- Systems

Coupling Values and Performance Info

- Function Performance
Composition Method: Example

- **FFT NBP Kernel**
  - Focus on the reverse FFT

![Diagram]

<table>
<thead>
<tr>
<th>Kernel Pair</th>
<th>Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>cffts1-trans.</td>
<td>1.0</td>
</tr>
<tr>
<td>trans-cffts2</td>
<td>1.0</td>
</tr>
<tr>
<td>cffts2-cffts1</td>
<td>1.0</td>
</tr>
<tr>
<td>cffts1-cffts1</td>
<td>1.0</td>
</tr>
</tbody>
</table>
## Results

- **FFT NPB on SGI Origin**

<table>
<thead>
<tr>
<th></th>
<th>2 proc.</th>
<th>4 proc.</th>
<th>8 proc.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S</strong></td>
<td>4.81s</td>
<td>2.4s</td>
<td>1.23s</td>
</tr>
<tr>
<td></td>
<td>5.29s</td>
<td>2.64s</td>
<td>1.32s</td>
</tr>
<tr>
<td><strong>W</strong></td>
<td>11.36s</td>
<td>5.56s</td>
<td>2.81s</td>
</tr>
<tr>
<td></td>
<td>10.95s</td>
<td>5.47s</td>
<td>2.74s</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>208.44s</td>
<td>103.43s</td>
<td>52.89s</td>
</tr>
<tr>
<td></td>
<td>198.29s</td>
<td>99.15s</td>
<td>49.57s</td>
</tr>
</tbody>
</table>
Related Work

- **Automated Performance Prediction**
  - *Dimemas* (Hope, Girona)
  - *MVA* (Liang, Tripathi)
  - *MACS* (Davidson)
  - *Queing model* (Mak, Lundtrom)
  - *Abstract operations* (Saavedra-Barrera, Smith)

- **Performance Environments**
  - *PACE* (Kerbyson, Harper)
  - *POEMS* (Brown)
Summary

- Need performance modeling and analysis for distributed application
- Presented the automated modeling component of Prophesy
- Currently have three techniques automated:
  - Curve Fitting
  - Parameterization
  - Composition
- Move toward developing a composition algebra

URL: prophesy.mcs.anl.gov
Future Work

- *Increase the number of kernels analyzed*
- *Continue exploring the composition algebra*
- *Consider what happens with code between loops and functions*
**Prophesy:** prophesy.mcs.anl.gov
Prophesy Registration

NOTE: Register your PAIDE application
This is done only once per executable.

Download PAIDE System

The PAIDE system description:
The PAIDE (Prophesy Automatic Instrumentation and Data Entry) system is the data collection component of the Prophesy system, with the goal of minimizing instrumentation overhead and code. It focuses on the automatic instrumentation of codes at the level of basic blocks, procedures, or functions. The default mode consists of instrumenting the entire code at the level of basic loops and procedures. A user can specify that the code be instrumented at a finer granularity than that of loops or identify the particular events to be instrumented. The resultant performance data is automatically placed in the performance database and is used by the data analysis component to produce an analytical performance model with coefficients, at the granularity specified by the user.

home
Prophesy Data Entry

Data Entry for Users

User Name: xxx
Password: ****

Submit Reset

Click Forgot Password, we will e-mail password to you.

Click Change Password to change your password.

home back
Prophesy Data Entry

The existing application list for the user is as follows:

<table>
<thead>
<tr>
<th>Application Name</th>
<th>Version</th>
<th>Description</th>
<th>Email</th>
<th>Owner Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAS Parallel CG Benchmark</td>
<td>2.30</td>
<td></td>
<td><a href="mailto:wuxf@ece.nwu.edu">wuxf@ece.nwu.edu</a></td>
<td>wuxf</td>
</tr>
<tr>
<td>NAS Serial CG Benchmark</td>
<td>2.30</td>
<td></td>
<td><a href="mailto:wuxf@ece.nwu.edu">wuxf@ece.nwu.edu</a></td>
<td>wuxf</td>
</tr>
<tr>
<td>NAS Serial IS Benchmark</td>
<td>2.30</td>
<td></td>
<td><a href="mailto:wuxf@ece.nwu.edu">wuxf@ece.nwu.edu</a></td>
<td>wuxf</td>
</tr>
<tr>
<td>NAS Parallel IS Benchmark</td>
<td>2.30</td>
<td></td>
<td><a href="mailto:wuxf@ece.nwu.edu">wuxf@ece.nwu.edu</a></td>
<td>wuxf</td>
</tr>
</tbody>
</table>

Must choose one to view it next time: Run Module

New Application Data Entry:
Note: The fields with * must be filled out.

*Application Name: 

*Application Version: 

Application Description: wuxf

*User Name: wuxf
*Email: wuxf@ece.nwu.edu

Must choose one to add it next time: Run Module

Submi| Reset
# Prophesy Data Entry

The existing executable list is as follows.

<table>
<thead>
<tr>
<th>Executable Name</th>
<th>Description</th>
<th>Application Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>cg_A_8 sgi</td>
<td></td>
<td>NAS Parallel CG Benchmark</td>
</tr>
<tr>
<td>cg_A_4 sgi</td>
<td></td>
<td>NAS Parallel CG Benchmark</td>
</tr>
<tr>
<td>cg_A_2 sgi</td>
<td></td>
<td>NAS Parallel CG Benchmark</td>
</tr>
</tbody>
</table>

Must choose one to view it next time: System Application_Performance Input

**New Executable Information**

Note: The fields with * must be filled out.

*Executable Name:*

*Application Name: NAS Parallel CG Bench

Executable Description:

Must choose one to add it next time: System Application_Performance Input

**Go back to Prophesy Home**
Prophesy Data Query

Select the item(s) to be added to the search. Depress either the "SHIFT" key or the "CONTROL" key in order to select multiple items. After selecting, please click the column button which is at the top of the items to submit.
Prophesy Data Query

Prophesy Database Query

Applications

Executables

Systems

Pure Functions

NAS Serial CG Benchmark
NAS Parallel IS Benchmark

cg_A_sgl

is_A_8_sgl

is_A_4_sgl

is_A_2_sgl

ECE SGI Origin2000

mv_mult

.apps= (NAS Parallel IS Benchmark) AND .execs= (is_A_0_sgl;is_A_4_sgl)

AND .sys= (ECE SGI Origin2000)

Name: [Query1] Save Clear

The text window can be edited, if desired.

{query1 = [ apps= (NAS Parallel IS Benchmark) AND execs= (is_A_0_sgl;is_A_4_sgl) AND sys= (ECE SGI Origin2000) ]}

List Combining: OR AND

Submit Query Reset Query Home
**Prophesy Data Query**

**Query Results**

*Applications: NAS Parallel IS Benchmark*

*Executables: is_A_8_sgi; is_A_4_sgi*

*Systems: ECE SGI Origin 2000*

---

**Note:** Choose what kinds of query attributes you want to show the results.

<table>
<thead>
<tr>
<th></th>
<th>Runtime (1)</th>
<th>Functions (2)</th>
<th>Modules (3)</th>
<th>Systems (4)</th>
<th>Compilers (5)</th>
<th>Number of Processors (6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime (1)</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>Functions (2)</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>Modules (3)</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>Systems (4)</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
</tr>
<tr>
<td>Compilers (5)</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
</tr>
<tr>
<td>Number of Processors (6)</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
</tr>
</tbody>
</table>

---

*Go back to Prophesy Home*