Performance Tools – Thoughts and Experiences

Seetharami R. Seelam
The University of Texas - El Paso
Computer Science
seelam@cs.utep.edu
Outline

- Issues facing performance analysis research today and in the future
- What should the community do?
  - Standard APIs
  - Smart data collection and analysis
  - Dynamic filtering and instrumentation
- Steps to address some of the issues that performance analysis is facing
Outline

- Issues facing performance analysis research today and in the future
- What should the community do?
  - Standard APIs
  - Smart data collection and analysis
  - Dynamic filtering and instrumentation
- Steps to address some of the issues that performance analysis is facing
Performance Analysis:
Current and Future Issues - 1

- Performance problems are vaguely defined
- Fine-grained instrumentation generates GBs of data; coarse-grained instrumentation loses important information
- Appropriate sampling frequency must be determined; use hints from application programmers
- Correlation of GBs of data to program and system components is difficult
Performance Analysis:
Current and Future Issues - 2

- Providing checks and balances seems to be a difficult problem
- Scalability of tools is a big concern and it is growing
- Adaptivity in application, operating system or architecture increases complexity of related issues
- Poor communication with application developers exacerbates problems
Outline

- Issues facing performance analysis research today and in the future
- What should the community do?
  - Standard APIs
  - Smart data collection and analysis
  - Dynamic filtering and instrumentation
- Steps to address some of the issues that performance analysis is facing
What should the community do?

- Provide standard APIs
- Collect relevant performance data
- Collect data via data-centric instrumentation
- Provide means for analysis
- Automatically identify relevant performance data and bottlenecks
Provide Standard APIs

- Well-defined interfaces (e.g., POMP)
- Ability to tightly control fine-grained instrumentation
- Ability to invoke instrumentation per function, per thread, per process, per system
- Instrumentation based on flow of data, as well as flow of program control
- Provision of profile and trace libraries
Collect Relevant Performance Data – Scalability Tradeoffs

- Profiles (aggregate counts)
- Complete event traces
- Aggregate counts by function

Less storage
Less overhead
Lower level of detail

More storage
More overhead
Higher level of detail
Data Collection via Data-centric Instrumentation

- Control-centric performance information
- Data-centric performance information
  - Understanding precise memory references is crucial to efficient memory hierarchy utilization
  - Example: c2c identifies performance problems not with control flow but with respect to flow of information through data structures
Example: Analysis of CG code

CG code segment with a large number of cache line transfers

```c
1 !$omp do
2   do j=1,naa+1
3     ...
4       p(j) = r(j)
5     enddo
6   ...
7   !$omp do
8     do j=1,lastrow-firstrow+1
9       sum = 0.0
10      do k=rowstr(j),rowstr(j+1)-1
11       sum = sum + a(k)*p(colidx(k))
12     enddo
13     q(j) = sum
14   enddo
```

Figure 3. Event/Architecture Specification

80% total program cache transfers

<table>
<thead>
<tr>
<th>Array</th>
<th>Function</th>
<th>Memory References</th>
<th>Stores</th>
<th>Cold Misses</th>
<th>$c2c$ transfers</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>conj_grad@OL@B@OL@F</td>
<td>741241600</td>
<td>0</td>
<td>2624</td>
<td>1048576</td>
</tr>
<tr>
<td>a</td>
<td>conj_grad@OL@B@OL@F</td>
<td>741241600</td>
<td>0</td>
<td>16588</td>
<td>70057</td>
</tr>
<tr>
<td>z</td>
<td>conj_grad@OL@B@OL@13</td>
<td>29649664</td>
<td>0</td>
<td>2621</td>
<td>39427</td>
</tr>
<tr>
<td>z</td>
<td>conj_grad@OL@B@OL@11</td>
<td>11200000</td>
<td>5600000</td>
<td>0</td>
<td>2395</td>
</tr>
<tr>
<td>colidx</td>
<td>conj_grad@OL@B@OL@F</td>
<td>741241600</td>
<td>0</td>
<td>0</td>
<td>403</td>
</tr>
</tbody>
</table>

cache-to-cache transfers in NAS CG
Outline

- Issues facing performance analysis research today and in the future
- What should the community do?
  - Standard APIs
  - Smart data collection and analysis
  - Dynamic filtering and instrumentation
- Steps to address some of the issues that performance analysis is facing
Smart Data Collection & Analysis

- Leverage efficient storage techniques from databases
- Use compression as well as sampled event traces to reduce the amount of data
- Use multivariate statistical analysis to eliminate redundant data
- Utilize autonomous agents to dynamically capture and analyze relevant performance data, and “throttle” the amount and type of information collected
Smart Data Collection & Analysis - 2

PETrAT

Data Collection Environment
- TPC-C
- p690

Sampled Event Traces

Load DB Java Tool

Database

Report Generation Java Tool

Reports
- BufferPool 56893 29384
- Data, BSS, Heap 8799 4855
- Kernel 23485 9840

Microsoft Excel

Graphs

Performance Event Trace Analysis Tool
Diana Villa’s Master’s Thesis – Work with IBM-Austin
Smart Data Collection & Analysis - 3

Sampled Event Traces – Analysis & Results

Resolution of L2 Data Load Misses

Distribution of L3 Load Hits by Address Region – 32-processor p690

Distribution of L3 Data Load Hits by Cache line

IBM Petascale Tools Strategy Workshop
May 3-4, 2005
Smart Data Collection & Analysis - 4
Multivariate Statistical Analysis

Scatterplot /correlation matrix to identify redundant counters

High correlation – just measure one, predict others

Table 3: Scatterplot matrix for UMT on 288 tasks with raw data from 7 counters
Automatic Identification of Relevant Performance Data

- Autonomous agents that can activate/deactivate or throttle instrumentation based on possible performance problems
- 9.9% of the data contains 89.1% of the problems
- Agents that can map between data flow and control flow of information
- Leverage dynamic adaptation techniques from OS or application layers
More Observations

- We need thermometers and stethoscopes
- X-rays and MRIs are what we have
  - Programmers need to know why the bone is broken
  - We are only showing that the bone is broken
- We need automatic tools to analyze the information
- Finding methods that discard useless information is critical
Addressing the Scalability Challenges of Performance Data Collection & Analysis

- Standardization of APIs
- Scalable data collection
- Throttled data collection
  - Dynamically adapt w.r.t. data being collected
    - Type and amount
- Adaptive instrumentation and data collection
- Use of off- and on-line analysis to adapt heuristics that trigger adaptations
- Autonomous agents
- Scalable analysis of trace data (database queries, EARL modules)
- Scalable visualization of analytic results
Acknowledgements

- David Klepacki and the IBM team that made this workshop a reality
- My mentors and collaborators at IBM: Simone Sbaraglia, Eknath Kattamuri, and Luiz Derose (now at Cray)
- Collaborators from IBM-Austin
- My Ph.D. advisor, Patricia Teller
References


