Knowledge Representation for (Computer Programs) Performance Analysis

Anton Dergunov

Lobachevsky State University of Nizhni Novgorod

Summer School “Semantic Web - Ontology Languages and Their Use”
Technische Universität Dresden
August 18 - 31, 2013
MPI (Message Passing Interface)

Legend:
- user code
- MPI function calls
- message
Performance problems in MPI applications

Synchronized send-receive

Late sending problem

Late receiving problem
Late broadcast performance problem
Early reduce performance problem
Performance improvement tasks

1. Select metrics
2. Collect
3. Visualize
4. Detect performance problems
5. Search for a way to fix
6. Fix
## Existing Systems

<table>
<thead>
<tr>
<th>Select metrics</th>
<th>Intel Trace Collector and Analyzer, Jumpshot</th>
<th>SCALASCA, Aksum, etc.</th>
<th>Performance Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collect</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Visualize</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect performance problems</td>
<td></td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Search for a way to fix</td>
<td></td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Fix</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance Expert System

Preparation step (done by expert)

Usage step (done by user)

Knowledge Base

Trace rules

Analysis rules

Interpreter

Collector

Analyzer

Performance Expert System

Application

Trace

Detected problems and recommendations

Typical performance problem

Anton Dergunov

Knowledge Representation for (Computer Programs) Performance Analysis
Tracing and Analysis Workflow

Tracing

P1 → P2

Tracing rules → Events trace → Composite events construction rules → Composite events → Performance problem detection rules → Identified performance problems

Analysis

Problem detected!

Not problems detected.

Analysis rules
<functionPoint name="func_call_wait"
    pattern="^MPI_Waitall$">
    <event>

        <param name="requests" format="vector_of_int">

            <sizeSource>count</sizeSource>

            <itemsSource><![CDATA[
                reinterpret_cast<sint32_t*>(array_of_requests)
            ]]]></itemsSource>

    </param>

    <!-- ... -->

    </event>
</functionPoint>
Composite Events Construction Rules

declare composite event receive
  members(
    func_call_non_blocking_receive as e,
    shared func_call_wait as wait)
when
  member(e.request, wait.requests) and
  e.@process == wait.@process and
  e.@time < wait.@time
parameters(
  start_time = e.@time,
  wait_start_time = wait.@time,
  wait_finish_time = wait.@time + wait.@duration,
  source = e.source,
  dest = e.dest,
  tag = e.tag,
  comm = e.comm);
Performance Problem Detection Rules

declare problem for point_to_point
when
  send_start_time > recv_wait_start_time
parameters(
  name = "Late sending",
  description = "Sending message is initiated long after receive is initiated. As a result, blocking receive must wait.",
  advice = "Make changes in source code location " + recv_wait_call_site + ", so that receive happens after send is done",
  duration = send_start_time - recv_wait_start_time);
MPI Performance Problems Knowledge Base

- **Point-to-point:**
  - Late sending
  - Late receiving

- **Collective:**
  - Early reduce
  - Late broadcast
  - Wait at many-to-many
  - Wait at barrier

- **RMA (Remote Memory Access):**
  - Wait at window creation
  - Contention for access to remote memory
  - Late beginning of window exposure epoch
  - Early finish of window exposure epoch
Experiment: Heart Modelling Application

\[ v'(x,y) = f(v(x, y), \ v(x - 1, y), \ v(x + 1, y), \ v(x, y - 1), \ v(x, y + 1)) \]
## Experiment: Recommendations

<table>
<thead>
<tr>
<th>Problem</th>
<th>Recommendation</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Late sending</td>
<td>Synchronize send-receive operations...</td>
<td>28.02%</td>
</tr>
<tr>
<td>Early reduce</td>
<td>Initiate reduce later on root process...</td>
<td>24.16%</td>
</tr>
</tbody>
</table>
Experiment: Application Modification

- For each cell: calculate new values
- Exchange values on the border
Experiment: Application Modification

- For each cell: calculate new values
- For each cell on the border: calculate new values
- For each cell inside: calculate new values
- Exchange values on the border
Experiment: Application Modification

- For each cell: calculate new values
- For each cell on the border: calculate new values
- Initiate exchange of values on the border asynchronously
- For each cell inside: calculate new values
- Exchange values on the border
- Wait for exchange to finish
Experiment: Application Modification

- For each cell on the border: calculate new values
- Initiate exchange of values on the border asynchronously
- For each cell inside: calculate new values

- Wait for exchange to finish
Experiment: Performance Improvements

<table>
<thead>
<tr>
<th>PR</th>
<th>Before improvement</th>
<th>After improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>4936.51</td>
<td>4249.94</td>
</tr>
<tr>
<td>26</td>
<td>3757.66</td>
<td>2976.61</td>
</tr>
<tr>
<td>37</td>
<td>3057.43</td>
<td>2296.87</td>
</tr>
<tr>
<td>101</td>
<td>2055.38</td>
<td>1269.56</td>
</tr>
</tbody>
</table>
Future Work

- Extend knowledge base for other types of performance problems:
  - Other parallel libraries (OpenMP, etc.)
  - Microarchitectural issues (cache hierarchy, etc.)
- Machine learning approaches to automatically create tracing and analysis rules
Thank you!