Critical Path Profiling of Message Passing and Shared-Memory Programs

Written by Jeffrey K. Hollingsworth

Presented by Shomir Wilson
CMSC 714, Fall 2005
Outline

- Introduction to critical path profiling
- A definition of a critical path
- How to compute the critical path
- Critical path of partial program execution
- Message-passing implementation
- Shared memory implementation
- Conclusion
Introduction

Critical path profiling:
- is a way to identify performance bottlenecks in parallel programs.
- is more useful than sums of sequential metrics, such as CPU utilization.
- is computed *online*—during program execution—and for specific regions of a program.
- requires minimal memory and processing overhead to compute
The Critical Path

- Basically, the *critical path* of a program is *the longest, time-weighted sequence of events from the start of a program to its termination*.

- The length of a linear program’s critical path is simply the CPU time it consumes.

- In a parallel program, communication and synchronization events create multiple paths.
Definitions

- **Process**: A thread of execution with its own address space
- **Events**: Observable operations performed by a process (message passing: start-send, end-send, start-receive, end-receive; procedure calls; returns)
- **Program**: One or more processes that communicate (SPMD, MIMD, client-server)
- **Program execution (P)**: The total ordering of all events in all processes during a single execution of a program
- **Program execution trace**: A set of per-process logs of events
- **CPU time**: Per-process clock that runs when the process is not waiting for a message
The PAG and the Critical Path

Program Activity Graph (PAG): a graph of events in a single program trace. Nodes are events, and arcs impose an ordering on them. Arc labels indicate CPU time between events.

Restated, the critical path of a parallel program is the longest path through the PAG.
Online Critical Path Computation

- The user selects the procedures for profiling; generally the top 8-10 that (probably) consume the most CPU time.
- During execution, no additional messages are sent for critical path computation.
- An extra value is attached to each message sent, indicating the length of the longest path to the point of the send operation.
- When a message is received, that value is compared with the local length of the critical path. The larger of the two is stored.
- After computation, the longest path through any process is the critical path.
Each process stores:

- The longest path ending in the process
- The share of that path due to the selected procedure
- A flag indicating if the selected procedure is active
- The time of the process’ last recorded event
- The time of the last recorded event when the selected procedure was active
Critical Path of Partial Program Execution

- Computing the critical path for a selected fraction of a program is more complicated. The PAG must be divided into three disjoined sets of vertices: before, during, and after the selected fraction. However, the PAG is never explicitly constructed, and it is difficult to synchronize events without more communication.

- To end analysis before program termination (or compute an intermediate value), basically each process forwards its monitoring information to a single process that determines the maximum of given samples.
Online Critical Path Zeroing

- Logical zeroing is the reduction in the length of the critical path due to tuning specific procedures in it.
- Computing logical zeroing is similar to computing critical path length, except the “net” path length is compared at merge nodes.
- The “net” path length is the path length minus the share of the path due to the selected procedure.
Initial (Message Passing) Implementation

- An implementation of the online critical path algorithm was added to the Paradyn Parallel Performance Tools.
- The initial implementation was done using PVM. Instrumentation overhead was small.

<table>
<thead>
<tr>
<th>Number of CP Items</th>
<th>~75% Computation</th>
<th>~60% Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wall Time</td>
<td>Overhead</td>
</tr>
<tr>
<td>Base</td>
<td>154.1</td>
<td>1.9%</td>
</tr>
<tr>
<td>0</td>
<td>157.0</td>
<td>2.1%</td>
</tr>
<tr>
<td>1</td>
<td>157.4</td>
<td>2.2%</td>
</tr>
<tr>
<td>4</td>
<td>157.5</td>
<td>2.8%</td>
</tr>
<tr>
<td>8</td>
<td>158.4</td>
<td>2.9%</td>
</tr>
<tr>
<td>16</td>
<td>158.5</td>
<td>3.6%</td>
</tr>
<tr>
<td>32</td>
<td>159.6</td>
<td>3.6%</td>
</tr>
</tbody>
</table>
Benchmarking Results

NAS Integer sort was run on eight nodes of an IBM SP-2 using the High Performance Switch for messaging.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>CP</th>
<th>% CP</th>
<th>CPU</th>
<th>% CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>nas_is_ben</td>
<td>8.8</td>
<td>31.7</td>
<td>103.1</td>
<td>67.8</td>
</tr>
<tr>
<td>do_rank</td>
<td>0.5</td>
<td>1.8</td>
<td>29.9</td>
<td>19.7</td>
</tr>
<tr>
<td>create_seq</td>
<td>18.5</td>
<td>66.5</td>
<td>18.7</td>
<td>12.3</td>
</tr>
</tbody>
</table>

Note that create_seq is only 12% of the overall execution time but over 66% of the critical path.

Benchmarking data on TSP and the GFDL Modular Ocean Model showed similar differences between CPU time and critical time.
Support for Shared Memory Programs

In shared memory programs, locks and barriers provide synchronization between threads of execution. The PAG is constructed around them.
More Benchmarking Results

<table>
<thead>
<tr>
<th>Procedure</th>
<th>CPU</th>
<th>CP</th>
<th>CP Zero</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Percent</td>
<td>Time</td>
</tr>
<tr>
<td>IntersectHuniformPrintlist</td>
<td>44.5</td>
<td>54.1</td>
<td>12.0</td>
</tr>
<tr>
<td>Shade</td>
<td>28.9</td>
<td>35.1</td>
<td>9.1</td>
</tr>
<tr>
<td>ReadGeoFile</td>
<td>4.5</td>
<td>3.5</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Fig. 12. CP vs. traditional profiling for the raytrace application. The results are for computing Prof, CP, and CP Zero for each three procedures. Both CP and CP Zero assign a significantly higher importance to ReadGeoFile than CPU profiling does.

The Raytrace application (from the Splash II benchmark suite) was run on a DEC Alpha shared memory system. Note that ReadGeoFile is about four times more important than the CPU metric shows.
Conclusion

Critical path profiling:

- provides a useful metric to search for bottlenecks in parallel programs.
- has a relatively small impact on program performance during measurement.
- can be used in both message-passing and shared memory environments.