Enhanced Performance Analysis of Multi-core Applications with an Integrated Tool-chain

Parallel Programming for Multi-core Architectures

www.parma-itea2.org

ParCo 2009 – ParMA Symposium

Thomas William
Agenda

- KOJAK / Scalasca toolset
- Vampir tool environment
- Vampir D-Bus interface
- KOJAK ➔ Vampir integration

- INDEED application

- Optimization
KOJAK / Scalasca toolset

- Open source
- Jülich Supercomputing Centre

- Automatic performance analysis
  - Instrument C, C++, and Fortran parallel applications
    - Based on MPI, (simple) OpenMP, or hybrid
  - Collect call-path-profiles and EPILOG event traces
  - Scan trace for event patterns representing inefficiencies
    - **KOJAK**: sequential analysis
    - **Scalasca**: parallel analysis
  - Categorize and rank inefficiencies found
  - Visualize via **CUBE** browser
Example: KOJAK analysis of GNS code

Identified Problems

Where in source?

Which processes are affected?

WaitAtNxN Pattern:
- collective OP
- collective OP
- collective OP

William, Mix, Mohr, Menzel, Voigtländer
Vampir tool environment

- **VampirTrace**
  - Open source
  - Event trace measurement system
  - Instruments C, C++, and Fortran
    - MPI, (simple) OpenMP, or hybrid, I/O
  - Collects event traces in OTF format
- **Post-mortem visual analysis**
  - Developed originally by Jülich and since 1997 by ZHR/ZIH of TU Dresden
  - Commercial distribution by GWT-TUD
  - **Vampir**: sequential visualizer
  - **VampirServer**: distributed client / parallel server architecture
Example: Vampir display of GNS code

- **Color** indicates state of process
- **Messages**
- **Collective Operations**

**Vampir - Timeline**
- gns.cff (0.0 s - 11.982 s = 11.982 s)
- Time轴：0.0 s to 11.982 s
- Processes (0-3)

**Vampir - Message Statistics**
- Detailed Process Timeline
- Message Statistics

William, Mix, Mohr, Menzel, Voigtländer
Vampir D-Bus interface

Object Paths

Interfaces

com.gwt.vampir

Methods

- ConnectToServer(String serverURL, Int32 port) → (String statustext)
- Exit()
- Hello(String name) → (String greeting)
- OpenDisplay(String DisplayName) → (Boolean status)
- OpenTrace(String filename) → (String statustext)
- ZoomDisplay(String DisplayName, Double StartTick, Double EndTick, String TimeFormat) → (Boolean status)

Signals

- SanityCheckFailed(String)
- SanityCheckOK(String)
- TraceLoadFinished(String)
- TraceLoadProgress(Int32)

org.freedesktop.DBus.Introspectable
KOJAK ➔ Vampir integration
KOJAK ➔ Vampir integration III
KOJAK ➔ Vampir integration IV

William, Mix, Mohr, Menzel, Voigtländer
INDEED application

• numerical simulation of metal forming processes, e.g.
  – roll forming
  – crash forming
  – hydro forming of tubes and welded blanks
  – main application: (hydro mechanical) deep drawing
Main characteristics:
- implicit finite element software
- specially developed thick shell elements
- adaptive mesh refinement accounting for tool radii and local blank curvature
- new parallel version based on a domain decomposition approach is currently under development
Parallelization results in new challenges for a rezoning step:
- safeguard correct element connectivity across adjacent sub-domains
- update information concerning neighboring sub-domains
- take into account a repartitioning of the blank if the computational load becomes too imbalanced
Original version in Cube3
Max. severity shown in Vampir
Max. severity shown in Vampir II

Location: process 0
Operation: find_neighbour_by_coor (969)
Activity: USR (209)
Interval: 2:20.067 - 2:20.231
Duration: 0.164 s
Source: filter_gvirtop.f90 : 25
Next Activity: 2:20.234 - 2:20.234
Previous Activity: 2:20.067 - 2:20.067
Max. severity shown in Vampir III
Max. severity shown in Vampir IV

Participants: Process(s) 2-32
Operation: MPI_Scatter
Communicator: 20000000
Duration: 1.401 s
Findings:

- Scalasca shows time “wasted” in MPI_Scatter
- Integration with Vampir makes it possible to investigate reason for this bottleneck
  - Look at history leading to the bottleneck
  - Examine possible side affects
- Function `find_neighbour_by_coor` is running in only one process

Solution:

- Parallelizing `find_neighbour_by_coor` to eliminate bottleneck
Optimized version in Cube3
William, Mix, Mohr, Menzel, Voigtländer
Optimized version in Vampir

William, Mix, Mohr, Menzel, Voigtländer
Comparison of severity

William, Mix, Mohr, Menzel, Voigtländer
Vampir comparison of severity traces

<table>
<thead>
<tr>
<th>Process</th>
<th>Status</th>
<th>Time</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process 1</td>
<td>OK</td>
<td>0:00</td>
<td></td>
</tr>
<tr>
<td>Process 2</td>
<td>OK</td>
<td>0:01</td>
<td></td>
</tr>
<tr>
<td>Process 3</td>
<td>OK</td>
<td>0:02</td>
<td></td>
</tr>
<tr>
<td>Process 4</td>
<td>OK</td>
<td>0:03</td>
<td></td>
</tr>
<tr>
<td>Process 5</td>
<td>OK</td>
<td>0:04</td>
<td></td>
</tr>
<tr>
<td>Process 6</td>
<td>OK</td>
<td>0:05</td>
<td></td>
</tr>
<tr>
<td>Process 7</td>
<td>OK</td>
<td>0:06</td>
<td></td>
</tr>
<tr>
<td>Process 8</td>
<td>OK</td>
<td>0:07</td>
<td></td>
</tr>
<tr>
<td>Process 9</td>
<td>OK</td>
<td>0:08</td>
<td></td>
</tr>
<tr>
<td>Process 10</td>
<td>OK</td>
<td>0:09</td>
<td></td>
</tr>
<tr>
<td>Process 11</td>
<td>OK</td>
<td>0:10</td>
<td></td>
</tr>
<tr>
<td>Process 12</td>
<td>OK</td>
<td>0:11</td>
<td></td>
</tr>
<tr>
<td>Process 13</td>
<td>OK</td>
<td>0:12</td>
<td></td>
</tr>
<tr>
<td>Process 14</td>
<td>OK</td>
<td>0:13</td>
<td></td>
</tr>
<tr>
<td>Process 15</td>
<td>OK</td>
<td>0:14</td>
<td></td>
</tr>
<tr>
<td>Process 16</td>
<td>OK</td>
<td>0:15</td>
<td></td>
</tr>
<tr>
<td>Process 17</td>
<td>OK</td>
<td>0:16</td>
<td></td>
</tr>
<tr>
<td>Process 18</td>
<td>OK</td>
<td>0:17</td>
<td></td>
</tr>
<tr>
<td>Process 19</td>
<td>OK</td>
<td>0:18</td>
<td></td>
</tr>
<tr>
<td>Process 20</td>
<td>OK</td>
<td>0:19</td>
<td></td>
</tr>
<tr>
<td>Process 21</td>
<td>OK</td>
<td>0:20</td>
<td></td>
</tr>
<tr>
<td>Process 22</td>
<td>OK</td>
<td>0:21</td>
<td></td>
</tr>
<tr>
<td>Process 23</td>
<td>OK</td>
<td>0:22</td>
<td></td>
</tr>
<tr>
<td>Process 24</td>
<td>OK</td>
<td>0:23</td>
<td></td>
</tr>
<tr>
<td>Process 25</td>
<td>OK</td>
<td>0:24</td>
<td></td>
</tr>
<tr>
<td>Process 26</td>
<td>OK</td>
<td>0:25</td>
<td></td>
</tr>
<tr>
<td>Process 27</td>
<td>OK</td>
<td>0:26</td>
<td></td>
</tr>
<tr>
<td>Process 28</td>
<td>OK</td>
<td>0:27</td>
<td></td>
</tr>
<tr>
<td>Process 29</td>
<td>OK</td>
<td>0:28</td>
<td></td>
</tr>
<tr>
<td>Process 30</td>
<td>OK</td>
<td>0:29</td>
<td></td>
</tr>
<tr>
<td>Process 31</td>
<td>OK</td>
<td>0:30</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process</th>
<th>Status</th>
<th>Time</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process 1</td>
<td>OK</td>
<td>0:00</td>
<td></td>
</tr>
<tr>
<td>Process 2</td>
<td>OK</td>
<td>0:01</td>
<td></td>
</tr>
<tr>
<td>Process 3</td>
<td>OK</td>
<td>0:02</td>
<td></td>
</tr>
<tr>
<td>Process 4</td>
<td>OK</td>
<td>0:03</td>
<td></td>
</tr>
<tr>
<td>Process 5</td>
<td>OK</td>
<td>0:04</td>
<td></td>
</tr>
<tr>
<td>Process 6</td>
<td>OK</td>
<td>0:05</td>
<td></td>
</tr>
<tr>
<td>Process 7</td>
<td>OK</td>
<td>0:06</td>
<td></td>
</tr>
<tr>
<td>Process 8</td>
<td>OK</td>
<td>0:07</td>
<td></td>
</tr>
<tr>
<td>Process 9</td>
<td>OK</td>
<td>0:08</td>
<td></td>
</tr>
<tr>
<td>Process 10</td>
<td>OK</td>
<td>0:09</td>
<td></td>
</tr>
<tr>
<td>Process 11</td>
<td>OK</td>
<td>0:10</td>
<td></td>
</tr>
<tr>
<td>Process 12</td>
<td>OK</td>
<td>0:11</td>
<td></td>
</tr>
<tr>
<td>Process 13</td>
<td>OK</td>
<td>0:12</td>
<td></td>
</tr>
<tr>
<td>Process 14</td>
<td>OK</td>
<td>0:13</td>
<td></td>
</tr>
<tr>
<td>Process 15</td>
<td>OK</td>
<td>0:14</td>
<td></td>
</tr>
<tr>
<td>Process 16</td>
<td>OK</td>
<td>0:15</td>
<td></td>
</tr>
<tr>
<td>Process 17</td>
<td>OK</td>
<td>0:16</td>
<td></td>
</tr>
<tr>
<td>Process 18</td>
<td>OK</td>
<td>0:17</td>
<td></td>
</tr>
<tr>
<td>Process 19</td>
<td>OK</td>
<td>0:18</td>
<td></td>
</tr>
<tr>
<td>Process 20</td>
<td>OK</td>
<td>0:19</td>
<td></td>
</tr>
<tr>
<td>Process 21</td>
<td>OK</td>
<td>0:20</td>
<td></td>
</tr>
<tr>
<td>Process 22</td>
<td>OK</td>
<td>0:21</td>
<td></td>
</tr>
<tr>
<td>Process 23</td>
<td>OK</td>
<td>0:22</td>
<td></td>
</tr>
<tr>
<td>Process 24</td>
<td>OK</td>
<td>0:23</td>
<td></td>
</tr>
<tr>
<td>Process 25</td>
<td>OK</td>
<td>0:24</td>
<td></td>
</tr>
<tr>
<td>Process 26</td>
<td>OK</td>
<td>0:25</td>
<td></td>
</tr>
<tr>
<td>Process 27</td>
<td>OK</td>
<td>0:26</td>
<td></td>
</tr>
<tr>
<td>Process 28</td>
<td>OK</td>
<td>0:27</td>
<td></td>
</tr>
<tr>
<td>Process 29</td>
<td>OK</td>
<td>0:28</td>
<td></td>
</tr>
<tr>
<td>Process 30</td>
<td>OK</td>
<td>0:29</td>
<td></td>
</tr>
<tr>
<td>Process 31</td>
<td>OK</td>
<td>0:30</td>
<td></td>
</tr>
</tbody>
</table>
• Integration of Scalasca and Vampir via D-Bus:
  – Scalasca reveals bottleneck
  – Vampir adds temporal and spatial history
    → helps in finding a solution
  – Enhancement can be evaluated (Scalasca + Vampir)

Combination of tools results in information gain higher than the added gain of both tools each taken by itself
Reference

   H. Schoop. ZAMM 67. 4 : 237-239, 1987
   El Rifai Kassem, K. et al. (J.-L. Chenot, R. D. Wood. ed.).