Clustering and Alignment Methods for Structural Comparison of Parallel Applications

Scalable Tools Workshop 2016
Lake Tahoe, California, USA

Matthias Weber

August 1, 2016
Outline

- Motivation
- Structural Clustering of Processes
  - Determining Similarity
  - Efficient Computation and Storage of Clusters
  - Applicability Study
- Structural Alignment of Processes
  - Alignment of Multiple Process Sequences
  - Merged Call Graph
- Conclusion
Manual comparison of two processes: Default vs. optimized application run

- Manual comparison of process event streams is extremely challenging due to the large number of events and the need to correctly line up trace events
- Automatic support for event-wise trace comparison needed
Pairwise Structural Comparisons with Sequence Alignment Methods

- Sequence alignment allows to compare process structure in detail
- Pairwise comparisons expose differences between two processes
- The pairwise process comparison is computationally expensive, forbidding exhaustive comparison of all process combinations

[1]
AMG2006 – A parallel algebraic multigrid solver for linear systems - Comparison of the default version with an optimized version that performs less coarsening, avoiding a lot of expensive communication - Shown are the unaligned rank0 processes - Exact differences are hard to spot
AMG2006: Runtime analysis

- The optimized version runs faster and finishes about 1.25 seconds earlier
Outline

- Motivation
- **Structural Clustering of Processes**
  - Determining Similarity
  - Efficient Computation and Storage of Clusters
  - Applicability Study
- **Structural Alignment of Processes**
  - Alignment of Multiple Process Sequences
  - Merged Call Graph
- Conclusion
Structural Similarity Measure

- Structural information is contained in call trees (disregarding timing)
- Easily obtainable from call-path profiles or traces
- Differences between processes are based function pairs that represent the caller-callee relation:
  - \( \text{pairs}(P) := \{(F_1, F_2) : F_1 \text{ calls } F_2 \text{ on } P \text{ at least once}\} \)
- Measure is independent of: number of calls, number of iterations, recursion depth, timing
- Assumption: static executable \( \rightarrow \) increasing process count or problem size does not increase the number of function pairs

\[
\text{pairsim}(P_1, P_2) := \frac{|\text{pairs}(P_1) \cap \text{pairs}(P_2)|}{|\text{pairs}(P_1) \cup \text{pairs}(P_2)|}
\]

Function pairs of proc 1:

\[\text{pairs}(\text{proc 1}) = \{\epsilon \rightarrow \text{main}, \text{main} \rightarrow \text{init}, \] \[\text{main} \rightarrow \text{fopen}, \text{main} \rightarrow \text{fclose}, \] \[\text{init} \rightarrow \text{fopen}, \text{init} \rightarrow \text{fclose}\}\]

Function pairs of proc 2:

\[\text{pairs}(\text{proc 2}) = \{\epsilon \rightarrow \text{main}, \text{main} \rightarrow \text{init}, \] \[\text{init} \rightarrow \text{fopen}, \text{init} \rightarrow \text{fclose}\}\]

Function pair similarity of the two example processes:

\[\text{pairsim}(\text{proc 1}, \text{proc 2}) = \frac{4}{6}\]
Formal Context

- The function pair similarity measure is set-based and allows to use formal concept analysis methods
- Similarity data can be described as a formal context [4], which is a triple \((O, A, I)\), where
  - \(O\) is a set of objects,
  - \(A\) a set of attributes,
  - and \(I \subseteq O \times A\) an incidence relation associating objects with attributes
- Storing the information of all function pairs in a table is not scalable

Formal context:
\[ (\mathcal{P}, \mathcal{A} \subseteq \mathcal{F} \times \mathcal{F}, \mathcal{I}) \]

With:
\[
\begin{align*}
\mathcal{P} & := \{P_1, P_2, P_3, P_4\}, \\
\mathcal{F} & := \{F_1, F_2, F_3\}, \\
\mathcal{A} & := \{\epsilon \to F_1, F_1 \to F_2, F_1 \to F_3\}, \\
\mathcal{I} & := \{(P_1, \epsilon \to F_1), (P_2, F_1 \to F_2), (P_2, \epsilon \to F_1), \\
& \quad (P_2, F_1 \to F_3), (P_3, \epsilon \to F_1), (P_3, F_1 \to F_2), \\
& \quad (P_3, F_1 \to F_3), (P_4, \epsilon \to F_1), (P_4, F_1 \to F_3)\}
\end{align*}
\]

Incidence relation table for the example processes:

<table>
<thead>
<tr>
<th></th>
<th>(\epsilon \to F_1)</th>
<th>(F_1 \to F_2)</th>
<th>(F_1 \to F_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_1)</td>
<td>(\times)</td>
<td>(\times)</td>
<td></td>
</tr>
<tr>
<td>(P_2)</td>
<td>(\times)</td>
<td></td>
<td>(\times)</td>
</tr>
<tr>
<td>(P_3)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\times)</td>
</tr>
<tr>
<td>(P_4)</td>
<td>(\times)</td>
<td></td>
<td>(\times)</td>
</tr>
</tbody>
</table>
Four example processes:

- Concept lattices order and store formal contexts efficiently
- Similar processes are grouped during construction
- Lattices have a small memory footprint; each process and each function pair occurs exactly once in the lattice
- Lattice construction is done using the algorithm from van der Merwe [5], that allows iterative adding of processes
- Expected complexity for building and storing the lattice is linear, the worst case is complexity is exponential

Incidence relation table for the example processes:

<table>
<thead>
<tr>
<th></th>
<th>$\epsilon \to F_1$</th>
<th>$F_1 \to F_2$</th>
<th>$F_1 \to F_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$P_2$</td>
<td>$\times$</td>
<td></td>
<td>$\times$</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$P_4$</td>
<td>$\times$</td>
<td></td>
<td>$\times$</td>
</tr>
</tbody>
</table>
Concept lattice for BT with 16 MPI processes (red) and 15 OpenMP threads (blue) per process:

- 256 processes in total
- 3 groups
  - Two groups with MPI processes (red)
  - One group with OpenMP threads (blue)
- All processes share 56 function pairs
- No process executes all function pairs
Applicability Study

- Study using 15 HPC applications with different characteristics
- $t_{\text{eval}}$ denotes the time to construct the lattice containing all processes and to compute the similarity matrix
- For all applications except ParaDiS $t_{\text{eval}}$ is below 0.1 seconds
- For 10 applications the number of process groups is below 10

<table>
<thead>
<tr>
<th>Application</th>
<th>Processes</th>
<th>Func. pairs</th>
<th>$t_{\text{eval}}$ (ms)</th>
<th>Lattice Nodes</th>
<th>Process Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPL</td>
<td>2,360</td>
<td>8</td>
<td>&lt; 10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>GROMACS</td>
<td>36</td>
<td>1,381</td>
<td>&lt; 10</td>
<td>24</td>
<td>11</td>
</tr>
<tr>
<td>CCLM</td>
<td>180</td>
<td>180</td>
<td>&lt; 10</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>COSMO-SPECS</td>
<td>100</td>
<td>50</td>
<td>&lt; 10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>WRF</td>
<td>64</td>
<td>774</td>
<td>&lt; 10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>FD4</td>
<td>65,536</td>
<td>55</td>
<td>55</td>
<td>22</td>
<td>14</td>
</tr>
<tr>
<td>HOMME</td>
<td>1,024</td>
<td>179</td>
<td>&lt; 10</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>AMG2006</td>
<td>1,024</td>
<td>440</td>
<td>66</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>IRS</td>
<td>64</td>
<td>989</td>
<td>34</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>LULESH</td>
<td>432</td>
<td>406</td>
<td>49</td>
<td>182</td>
<td>35</td>
</tr>
<tr>
<td>ParaDiS</td>
<td>128</td>
<td>649</td>
<td>3,486</td>
<td>6,367</td>
<td>74</td>
</tr>
<tr>
<td>PIConGPU</td>
<td>39</td>
<td>474</td>
<td>11</td>
<td>60</td>
<td>17</td>
</tr>
<tr>
<td>BT-MZ</td>
<td>16</td>
<td>126</td>
<td>&lt; 10</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>HPCC MPI-FFTE</td>
<td>128</td>
<td>109</td>
<td>70</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>PEPC</td>
<td>16,384</td>
<td>113</td>
<td>15</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Outline

- Motivation
- Structural Clustering of Processes
  - Determining Similarity
  - Efficient Computation and Storage of Clusters
  - Applicability Study
- **Structural Alignment of Processes**
  - Alignment of Multiple Process Sequences
  - Merged Call Graph
- Conclusion
Structural Comparisons of Multiple Sequences

- Progressive multiple sequence alignment (MSA) can align many sequences to one alignment block
- Progressively applied pairwise alignments add new sequences to the alignment block
- Structural pre-clustering helps to select processes for comparison
- MSA allows to compare all processes of a cluster in detail
Hierarchical Multiple Sequence Alignment Approach

- Aligning full process sequences is too computationally expensive
- The hierarchical approach exploits the call-tree structure, and splits up process sequences into several smaller sub-sequences
### Step 1

#### Merged Process-Tree

- **proc 1/2/3**
- **m**
- **a**
- **b**
- **d**
- **c**

#### Multiple Sequence Alignments

- **proc 1**
  - **m**
- **proc 2**
  - **m**
- **proc 3**
  - **m**

### Step 2

#### Merged Process-Tree

- **proc 1/2/3**
- **m**
- **a**
- **b**

#### Multiple Sequence Alignments

- **proc 1**
  - **a**
  - **b**
  - **c**
- **proc 2**
  - **a**
  - **b**
  - **c**
- **proc 3**
  - **a**
  - **b**
  - **c**

### Step 3

#### Merged Process-Tree

- **proc 1/2/3**
- **m**
- **a**
- **b**
- **c**

#### Multiple Sequence Alignments

- **proc 1**
  - **a**
  - **b**
  - **c**
- **proc 2**
  - **a**
  - **b**
  - **c**
- **proc 3**
  - **a**
  - **b**
  - **c**
The hierarchical MSA method computes a merged call graph
- The merged call graph:
  - Contains the structural information of all processes
  - Highlights structural similarities and differences
  - Useful for subsequent performance analyses
  - Useful for scalable visualization of performance data
Merged Call Graph Example: AMG2006

- Merged call graph contains information of 64 processes
- White/gray parts are similar between all processes
- Colored areas indicate “missing” processes (GAP states)
- The color indicates the number of processes contained in the function:
  - Red: many processes
  - Blue: few processes
Conclusion

- Introduced a novel grouping method based on the structure of processes
  - Applicable for most application types
  - Grouping can be efficiently stored and computed
  - In most cases linear time complexity
  - In many cases the number of generated clusters remains low and stable for increasing process counts
  - Useful as pre-clustering step to improve the effectiveness of traditional analysis techniques
- Introduced a hierarchical multiple sequence alignment approach to compare the structure of processes
  - Compares the function call structure in detail
  - Merged call graph combines the complete structural information of multiple processes and highlights differences


