Structural Comparison of Parallel Applications

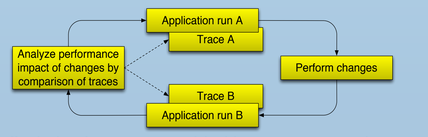
Matthias Weber[1], Kathryn Mohror[2], Martin Schulz[2], Holger Brunst[1], Bronis R. de Supinski[2], Wolfgang E. Nagel[1]

1. Technische Universität Dresden, 2. Lawrence Livermore National Laboratory

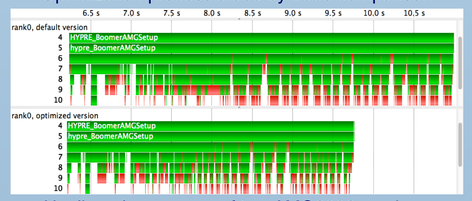
Event traces are a powerful tool for analyzing application performance. For example, they can expose performance differences between versions of code. However, manual comparison of traces is extremely challenging and time consuming because of the large volume of detailed data and the need to correctly line up trace events. Our solution is a set of techniques that automatically align traces so they can be compared, along with novel metrics that quantify the differences between traces, both in terms of differences in the event stream and timing differences across events. Additional clustering allows quick identification of similar or irregular traces across multiple runs of large applications.

Introduction:

- Highly detailed event traces are useful for performance optimization, they can be used to highlight differences across code versions
- Manual trace comparison is extremely challenging due to the large number of events and the need to correctly line up trace events
- No tool supports automatic comparison of event traces
- Our goal is to automatically detect and highlight differences between executions and to provide metrics that facilitate understanding of sources of the differences



Optimization process aided by trace comparison

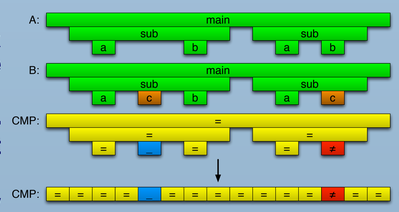


Unaligned processes of two AMG2006 versions

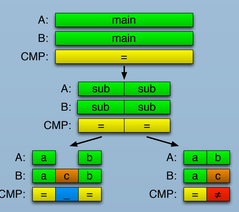
Alignment Algorithm:

Hierarchical Alignment (HA)

- To compare two processes we use a hierarchical approach that splits up the trace into several smaller sub-sequences
- Depending on their length, the sub-sequences are either aligned with a fast heuristic or using dynamic programming
- The HA algorithm has linear time complexity with respect to the complete trace length



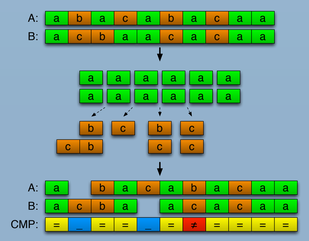
Comparison of two processes using HA



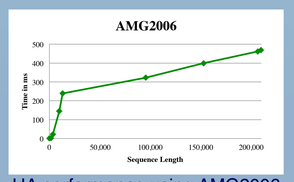
HA algorithm scheme

Alignment of Long Sub-sequences

- In case of long sub-sequence lengths (>10000) we apply a fast heuristic, working with linear time complexity
- The heuristic selects the most frequent function of both sequences as anchors and directly aligns all occurrences. The remaining functions between the anchors build new sub-sequences for further alignment



Heuristic scheme for alignment of long sub-sequences

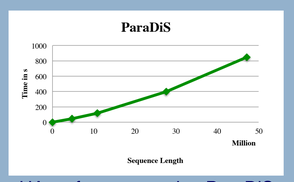


HA performance using AMG2006

Alignment of Short Sub-sequences

- Short sub-sequences (<10000) are aligned using dynamic programming
- DP has quadratic time complexity

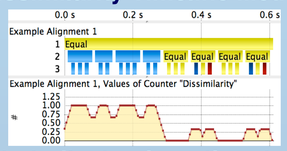
Alignment short sub-seqeunces



HA performance using ParaDiS

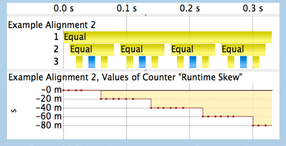
Comparison Metrics:

Dissimilarity Timeline Metric



This metric indicates how the similarity between two traces changes over time.

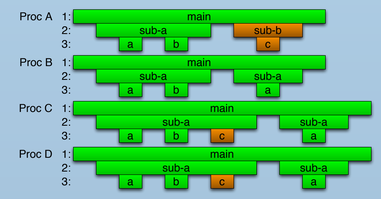
Runtime Skew Timeline Metric

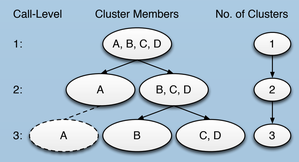


This metric facilitates understanding of the relative timing behavior across aligned traces.

Structural Clustering:

We cluster traces by their structure using a hierarchical approach that iteratively adds call-levels.
The computed clusters, groups of structurally identical traces, are highly suitable for further analysis or alternative clustering methods.



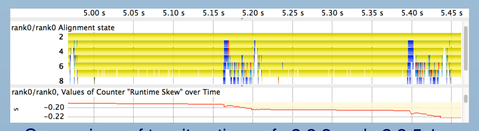


Case Studies:

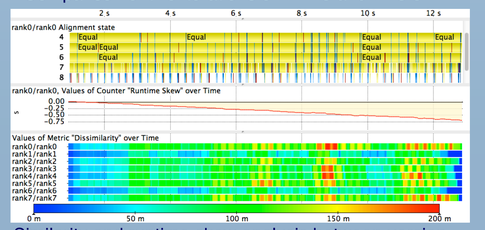
We demonstrate our trace comparison on AMG2006, a parallel algebraic multigrid solver, and ParaDiS, a code simulating crack propagation in materials.

AMG2006 – Comparison of the default with an optimized version. The optimized version performs less coarsening, avoiding a lot of expensive communication.

ParaDiS – Comparison of v2.2.3 and v2.3.5.1. v2.3.5.1 includes bug fixes, improvements, corrections, and advanced load balancing.



Comparison of two iterations of v2.2.3 and v2.3.5.1

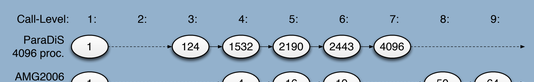


Similarity and runtime skew analysis between version 2.2.3 and 2.3.5.1 of ParaDiS

Similarity and runtime skew analysis between the default and optimized AMG2006 versions

Structural Clustering of AMG2006 and ParaDiS

With rising call-level both applications start to build clusters. In the highest call-levels no structurally identical traces exist.
Future work: Evaluate individual clusters, apply additional clustering techniques as second step to structurally identical clusters.



Number of clusters by call-level

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-POST-641577