# Trace comparison using a sequence alignment algorithm

Alef Farah, Lucas Mello Schnorr
PPGC, Institute of Informatics, UFRGS
Bento Gonçalves Avenue, 9500, Porto Alegre, RS, Brazil

## Abstract

*The comparison of execution traces enables an in depth analysis of the differences between two versions or runs of a parallel application. The automatic comparison of possibly huge trace files poses challenges on automation, scalability, and data visualization. In this article we present a strategy to overcome these difficulties based on a sequence alignment algorithm used in bioinformatics. We find the development of a scalable tool based on this strategy to be plausible. However, the initial experiments we executed led us to question if the technique in itself provides meaningful information for the analyst.*

## 1. Introduction

Application tracing and trace visualization enable the developer of parallel and distributed applications to examine his software in detail, in order to perform the fine-tuning often necessary while developing for complex environments. Visualization techniques coupled with data aggregation methods also allow an overview of the application behavior [2], which is often desired when the amount of registered events is overwhelming.

The developer might also wish to compare different versions of his application to identify performance loss or improvement within the modified parts of his program. It is also desirable to evaluate an application behavior in different architectures or networks, and comparing its execution traces is one way to identify both subtle and noticeable differences. The standard approach still is a manual comparison of the visualization of the trace files, leaving the pattern recognition work for the human brain. With the growth both in size and complexity of data and code, an automatic comparison is often not only desirable, but imperative.

Among the difficulties of an automatic comparison is the identification of similar regions between two programs with different execution times for each function call, possibly different order for asynchronous tasks and communications, and so forth. Performing the visualization of an execution trace in a scalable way is no trivial task with current data volumes. Besides being scalable performance-wise, the application also needs to render a comprehensible visualization for the user even with huge execution traces.

We tested proposed solutions for the aforementioned difficulties based on sequence alignment and data aggregation techniques. We implemented the former and experimented with the latter to evaluate their usefulness and feasibility, seeking possible improvements.

The rest of this paper is organized as follows. In section 2 we present the strategies we studied. In section 3 we present an overview of their implementation, and in sections 4 and 5 we show initial results on their usefulness and feasibility, respectively. Section 6 presents a summary of our contributions.

## 2. Related work

P. Velho, L. M. Schnorr, A. Legrand and H. Casanova [10] use markers to identify comparable regions, a strategy present in various tools such as the popular Vampir visualization tool [1]. However, this approach as it is currently used lacks automation as markers need to be added manually by the analyst. M. Weber et al [11] borrow a classic algorithm used in bioinformatics to do a pair-wise alignment of two trace files. They couple it with custom similarity metrics [12] and use a proprietary visualization tool [1] to display the results. We reproduced this strategy using free software tools such as Pajé [7], and evaluate the validity of our implementation in sections 4 and 5. Dosimont, Damien et al [2] use a data aggregation technique to summarize regions of large execution traces. We have experimented with aggregating data from two comparable trace files instead of aggregating comparable events within a single file, and present initial results in section 4.1.

In our work we studied the sequence alignment and data aggregation techniques, re-implementing the former and experimenting with the latter. Our objective was to reevaluate them and explore new possibilities, focusing on innovative visualization techniques for performance analysis of parallel applications.

## 3. Methodology

Aligning similar regions of two sequences allows the user to compare them more easily. More so, a similarity score can be calculated and used with visualization techniques to highlight these regions. This strategy, which belongs to a wide family of algorithms, is used to compare strings (edit distance, longest common subsequence), DNA sequences (Needleman-Wunsch [8], Smith-Waterman [9]), and in our case, sequence of events registered in two comparable trace files. The alignment of similar regions of the sequences can be thought of as shifting the elements in order to pair up those that are the same in both sequences.

To uniquely identify each event in the sequence we use its identifier from the trace file, the one already provided by the Pajé trace file format. For instance, for function calls the identifier is usually the function's name. Since execution times are different even between two executions of the same application in the same machine, event timestamp is not taken into consideration for comparison, only the order in which the events have occurred. This is the same idea used in the work in study [11].

To align the sequences we have used a modified version of Hirschberg's algorithm for finding the longest common subsequence between two strings [6], much like it is used in bioinformatics as a space linear version of the Needleman-Wunsch algorithm, used to align DNA sequences. The algorithm is quadratic in time; the procedure is as follows.

### 3.1. Implementation

While comparing the events in the sequence, we attribute the scores $\alpha$ for events that are equal, $\beta$ for events that differ, and $\gamma$ if an event is present only in one sequence, in which case we say a gap has occurred. The numerical value for each score can be defined by the user, the default being $\alpha = 2, \beta = -1, \gamma = -1$, penalizing differences and rewarding similarities. A scores matrix $D$ is recursively filled for every element in the sequences $A$ and $B$, which represent the sequence of events from matching execution flows from two trace files. This step is done for every execution flow present in both execution traces. One extra row and column are added at the beginning and initialized as follows. Let $\theta(i, j)$ be a function that returns $\alpha$ or $\beta$ comparing two events:

$$D(i,j) = \begin{cases} j * \gamma & i = 0 \\ i * \gamma & j = 0 \\ max \begin{cases} D(i-1, j-1) + \theta(i,j) \\ D(i, j-1) + \gamma \\ D(i-1, j) + \gamma \end{cases} & \begin{matrix} \\ i \neq 0, \\ j \neq 0 \end{matrix} \end{cases}$$

For each cell the score is calculated using the cell above it, to the left, and both (diagonal). Because of that, only two rows need to be kept in memory at any given time. This enables a space linear implementation of this algorithm.

This step is done recursively, dividing the problem into two smaller ones until a trivial case is reached, to which we apply the classical Needleman-Wunsch algorithm for aligning the sequences. The alignment is done by tracebacking in the matrix and finding a "shortest path" (i.e. passing through the cells with maximum similarity score) from the last cell to the first.

### 3.2. Parallelizing the algorithm

We first tried to parallelize the algorithm previously described to overcome its quadratic time. We use the call stack of the execution traces to determine which sub-sequences of events – i.e. which execution flows – can be compared in parallel. Since events from one execution flow are never comparable to events from different execution flows they can be aligned in parallel, as suggested in the work being studied [11].

Besides parallelization, a suggested optimization is to stop traversing a sub-tree when there is a difference in two events. In the green sub-sequences from Figure 1, events $3, 4$ and $C, D$ differ and their children would not be compared, but marked as different. This can be switched on or off by the user according to his needs. Notice that in Figure 1 there are events missing from the purple execution flow when compared with the matching execution flow from the other trace. In this case, events representing a gap would be added in order to fill the void.

## 4. Alignment visualization

We opted to use existing visualization tools and do so by writing the aligned sequences to new trace files. This allows the analyst to use whatever tool he desires, and also has the advantage that an alignment is only performed once. If the user wishes to revisit a previous alignment, he simply loads the already aligned trace files to his favorite visualization tool.

Figure 2 shows a Gantt chart of two unaligned execution traces. On top, the execution trace of an application parallelized with OpenMP using a "dynamic" scheduler, consisting of a loop calling a "do work" function. Below, the same application using a "guided" scheduler. Green polygons represent the "do work" events and red polygons the implicit barrier at the end of the loop. Events are displayed along the horizontal axis, which represents time. Each row represents a different execution flow. The Pajé trace visualization tool [7] was used to generate these representations. Manually comparing the visualizations it is clear that the

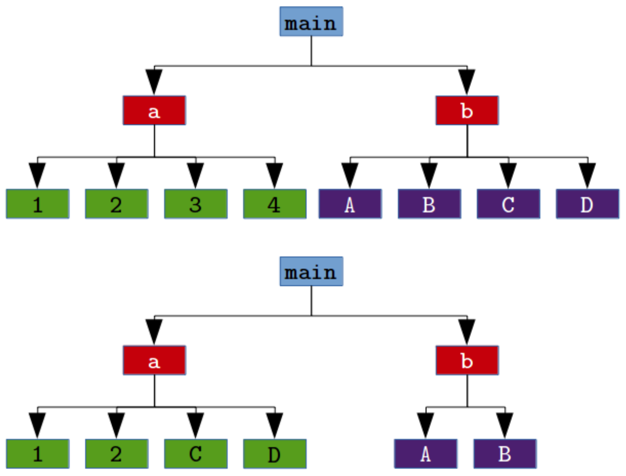

Figure 1: Visual representation of calling context trees for two trace files, where each polygon represents an event. The order in which the polygons appear in the figure (left to right) represent the order they occur in the trace file. Matching execution flows have the same color.

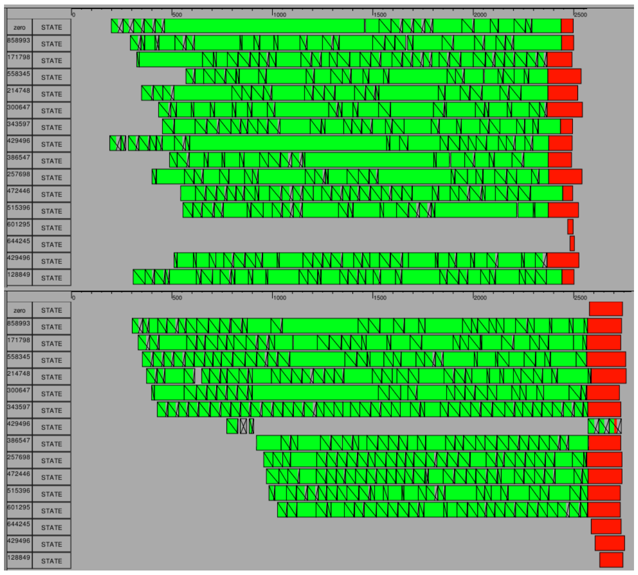

Figure 2: Unaligned execution traces.

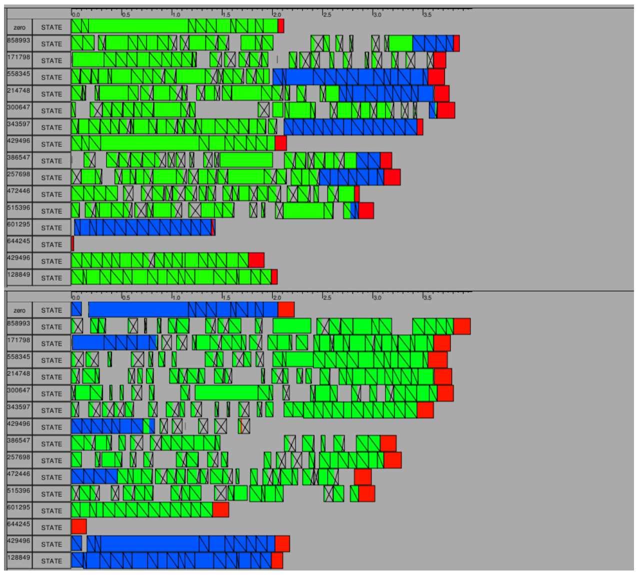

Figure 3: Aligned execution traces.

dynamic scheduler finished first and had a better distribution of the tasks among the threads.

Since during the alignment we disregard the timestamp from each event and insert gaps when one event is present only in one sequence, we end up loosing the original time information. In other words we shift events in time in order to align similar ones.

We visualized the alignment of simple execution traces and reproduce some results below. For the cases we studied, we found that the time shifting resulted in too much information loss, and we put the value of the shifted execution traces in question. For our test cases, we did not found that the alignment of similar regions provided enough benefits to justify the loss of timing information and meaning.

Figure 3 shows the aligned execution traces. Additional blue polygons represent gaps inserted due to an event being present solely in the other execution trace. Chronological information is lost so we can no longer easily determine which one had the shortest execution time. More so, we cannot easily determine how the tasks were distributed among the threads.

To overcome the issue with the loss of timing information, the authors of the original study propose several metrics and visualization techniques to indicate the shifts in time. Without them, we question the usefulness of the alignment strategy for certain cases. For the ones we studied, the manual comparison of the original trace visualizations proved to be better than the automatic approach.

## 4.1. Data aggregation

We also experimented with data aggregation techniques, more specifically with the Ocelotl Analysis Tool, which uses a custom spatiotemporal aggregation method [2]. We evaluated the possibility of comparing execution traces by aggregating regions that are homogeneous between them. At first, we merely merged two trace files into one and let the tool aggregate regions as it saw fit. Again, preliminary results shows that the information loss (this time due to the aggregation of incomparable regions) was too big, and the visual result of the merge was unsatisfactory.

An idea to overcome this issue is to merge the sequence alignment and data aggregation techniques. Because of the aforementioned difficulties with the stand-alone techniques, we postponed these tests until we resolve the issues we're having – possibly with similarity metrics and alternative visualization techniques.

## 5. Performance and scalability

We evaluated the performance of our application with artificial trace files of predetermined size generated by a script, and later with a few test cases [4]. Although more tests are necessary, our results were close to the ones obtained by previous works [11], both in total execution times and in the trend with which they grow as the file size increases – less than quadratic. Execution times for small files are seconds or less, and 2.5GB files with millions of events are aligned in less than two minutes in a personal computer [5].

In order to ascertain anything about the scalability of the tool, we have yet to test it with large scale traces and perform an in depth analysis. So far we obtained satisfactory results for small and medium trace files.

## 6. Conclusion

The preliminary tests we ran lead us to find that for the sequence alignment strategy a metric to indicate the time shifting is imperative. For our test cases, the alignment in itself does not seem to provide the analyst with enough information, and can cause confusion about the meaning of events shifted in time.

As to the performance and scalability of the alignment algorithm, our benchmarks show that it has reasonable execution times with medium trace files. The fact that the execution times rise in a less than quadratic way makes it plausible to think about a scalable tool based on this strategy.

We have some suggestions as to the alignment algorithm, such as using different alignment types, namely a semi-global alignment [3]. We are also working in a hybrid parallelization approach. However, until the visualization issues have not been solved, those suggestions will have to wait. Among the next steps in our research are additional test cases for the alignment visualization, specially with more complex applications, and the combination of the alignment and aggregation techniques. We are concurrently moving in a different direction and analyzing the intrusion of different tracing applications.

## 7. Acknowledgments

The authors would like to thank the National Counsel of Technological and Scientific Development (CNPq) for the financial support for the project.

## References

[1] H. Brunst, D. Hackenberg, G. Juckeland, and H. Rohling. Comprehensive performance tracking with vampir 7. In M. S. Müller, M. M. Resch, A. Schulz, and W. E. Nagel, editors, *Tools for High Performance Computing 2009*, pages 17–29. Springer Berlin Heidelberg, 2010.

[2] D. Dosimont, R. Lamarche-Perrin, L. Schnorr, G. Huard, and J.-M. Vincent. A spatiotemporal data aggregation technique for performance analysis of large-scale execution traces. In *Cluster Computing (CLUSTER), 2014 IEEE International Conference on*, pages 149–157, Sept 2014.

[3] S. R. Eddy. A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS computational biology*, 4(5):e1000069, 2008.

[4] A. Farah and L. M. Schnorr. Comparação de rastros para análise de desempenho de aplicações paralelas. *Anais da XV Escola Regional de Alto Desempenho, pp. 201-204*, 2015.

[5] A. Farah and L. M. Schnorr. Comparação de rastros para análise de desempenho de aplicações paralelas (slides). http://erad2015.inf.ufrgs.br/downloads/p/ic/s8-1.pdf, 2015.

[6] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM 18 (6): 341–343*, 1975.

[7] Kergommeaux, Stein, and Bernard. Pajé visualization tool. Parallel Computing, 26(10):1253–1274, 2000.

[8] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[9] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[10] P. Velho, L. M. Schnorr, H. Casanova, and A. Legrand. On the validity of flow-level tcp network models for grid and cloud simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 2013.

[11] M. Weber, R. Brendel, and H. Brunst. Trace file comparison with a hierarchical sequence alignment algorithm. In *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pages 247–254, July 2012.

[12] M. Weber, K. Mohror, M. Schulz, B. Supinski, H. Brunst, and W. Nagel. Alignment-based metrics for trace comparison. In F. Wolf, B. Mohr, and D. Mey, editors, *Euro-Par 2013 Parallel Processing*, volume 8097 of *Lecture Notes in Computer Science*, pages 29–40. Springer Berlin Heidelberg, 2013.