DiffTrace: Efficient Whole-Program Trace Analysis and Diffing

Saeed Taheri
staheri@cs.utah.edu
University of Utah
Salt Lake City, Utah, United States

Ian Briggs
ianbriggsutah@gmail.com
University of Utah
Salt Lake City, Utah, United States

Martin Burtscher
burtscher@cs.txstate.edu
Texas State University
San Marcos, Texas, U.S.A.

Ganesh Gopalakrishnan
ganesh@cs.utah.edu
University of Utah
Salt Lake City, Utah, U.S.A.

ABSTRACT

Abstract to be written

KEYWORDS

diffing, tracing, debugging

ACM Reference Format:
Saeed Taheri, Ian Briggs, Martin Burtscher, and Ganesh Gopalakrishnan. 2019. DiffTrace: Efficient Whole-Program Trace Analysis and Diffing. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Ganesh will write

I have some stuff in intro.tex (commented for now)

2 DFFTRACE COMPONENTS

2.1 General Idea

Here is a general overview of DiffTrace and its components

- Motivating example
- Problem statement
- Potential Approaches and Related Work
- Next subsections will explain the components that we have in our framework and the corresponding related work and background

2.2 Fault Injection

2.3 ParLOT

2-3 paragraph explanation about ParLOT and its mechanism [40]

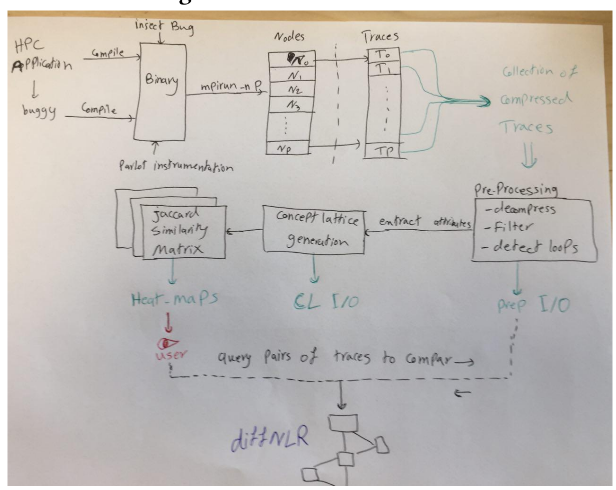
2.4 Filter

Include a table with all filters and their regular expressions

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Figure 1: diffTrace Overview



2.4.1 *General Filters.*

- Returns
- .plt
- Memory
- Network
- Polling
- String
- Customize
- IncludeEverything

2.4.2 *Target Filters.*

- MPI_
- MPIall
- MPI_ Collectives
- MPI_ Send/Recv
- OMPall
- OMPcritical
- OMPmutex

2.5 Nested Loop Recognition

2.5.1 *Background.*

2.5.2 *Implementation.*

2.6 Concept Lattice Analysis

2.6.1 Background.

- FCA (formal concept analysis) background and citations
- FCA applications in all areas
- FCA applications in Data Mining and Information Retrieval
- FCA applications in distributed systems (Garg's work)
- intro to Concept, Object, Attribute and other definitions

2.6.2 Objects/Attributes. Mapping of Object/Attribute (general) to Trace/Attribute (clTrace)

What do we expect to gain by doing so

- Single entity represents the whole execution of HPC application (can be used as signature/model in ML)
- Classifying similar behavior objects(traces)
- Efficient Incremental CL building makes it scalable
- Efficient full pair-wise Jaccard Similarity Matrix extraction

2.6.3 CL generation.

- background
- current approach

2.6.4 Jaccard Similarity Matrix.

- background
- LCA
- Benefits

2.7 diffNLR

- motivation
- diff algorithm
- visualization

2.8 FP-Trace

3 BACKGROUND AND RELATED WORK

3.1 STAT

Parallel debugger STAT[1]

- STAT gathers stack traces from all processes
- Merge them into prefix tree
- Groups processes that exhibit similar behavior into equivalent classes
- A single representative of each equivalence can then be examined with a full-featured debugger like TotalView or DDT

What STAT does not have?

- FP debugging
- Portability (too many dependencies)
- Domain-specific
- Loop structures and detection

3.2 Program Understanding

- Score-P [23]
- TAU [38]
- ScalaTrace: Scalable compression and replay of communication traces for HPC [35]
- Barrier Matching for Programs with Textually unaligned barriers [43]
- Pivot Tracing: Dynamic causal monitoring for distributed systems - Johnathan mace [29]
- Automated Charecterization of parallel application communication patterns [37]
- Problem Diagnosis in Large Scale Computing environments [31]
- Probablistic diagnosis of performance faults in large-scale parallel applications [25]
- detecting patterns in MPI communication traces - robert preissl [36]
- D4: Fast concurrency debugging with parallel differntial analysis - bozhen liu [28]
- Marmot: An MPI analysis and checking tool - bettina krammer [24]
- MPI-checker - Static Analysis for MPI - Alexandrer droste [14]
- STAT: stack trace analysis for large scale debugging - Dorian Arnold [1]
- DMTracker: Finding bugs in large-scale parallel programs by detecting anomaly in data movements [16]
- SyncChecker: Detecting synchronization errors between MPI applications and libraries - [10]
- Model Based fault localization in large-scale computing systems - Naoya Maruyama [30]
- Synoptic: Studying logged behavior with inferred models - ivan beschastnikh [4]
- Mining temporal invariants from partially ordered logs - ivan beschastnikh [6]
- Scalable Temporal Order Analysis for Large Scale Debugging - Dong Ahn [2]
- Inferring and asserting distributed system invariants - ivan beschastnikh - stewart grant [19]
- PRODOMETER: Accurate application progress analysis for large-scale parallel debugging - subatra mitra [32]
- Automaded : Automata-based debugging for dissimilar parallel tasks - greg [8]
- Automaded : large scale debugging of parallel tasks with Automaded - ignacio [26]
- Inferring models of concurrent systems from logs of their behavior with CSight - ivan [5]

3.3 Trace Analysis

- Trace File Comparison with a hierarchical Sequence Alignment algorithm [41]
- structural clustering : matthias weber [42]
- building a better backtrace: techniques for postmortem program analysis - ben liblit [27]
- automatically charecterizing large scale program behavior - timothy sherwood [39]

3.4 Visualizations

- Combing the communication hairball: Visualizing large-scale parallel execution traces using logical time - katherine e isaacs [21]
- recovering logical structure from charm++ event traces [20]
- ShiViz - Debugging distributed systems - [7]

Filter	Attributes	Top Process diffNLR Candidates	Top Thread diffNLR Candidates
11.mpi.cust.0K10	sing.log10	1:(25)r259.10699.0,(65)r265.20114.0 2:(20)r259.10698.0,(75)r265.20116.0 3:(0)r259.10694.0,(55)r265.20112.0	1:(2)r259.10694.2,(78)r265.20116.3 2:(17)r259.10697.2,(64)r265.20113.4 3:(49)r265.20110.4,(58)r265.20112.3
11.mpi.cust.0K10	doub.orig	1:(25)r259.10699.0,(65)r265.20114.0 2:(20)r259.10698.0,(75)r265.20116.0 3:(0)r259.10694.0,(55)r265.20112.0	1:(32)r259.10700.2,(54)r265.20111.4 2:(14)r259.10696.4,(74)r265.20115.4 3:(53)r265.20111.3,(53)r265.20111.3
11.mpicol.cust.0K10	doub.orig	1:(25)r259.10699.0,(65)r265.20114.0 2:(20)r259.10698.0,(75)r265.20116.0 3:(0)r259.10694.0,(55)r265.20112.0	1:(32)r259.10700.2,(54)r265.20111.4 2:(14)r259.10696.4,(74)r265.20115.4 3:(53)r265.20111.3,(53)r265.20111.3
01.mpicol.0K10	doub.orig	1:(25)r259.10699.0,(65)r265.20114.0 2:(20)r259.10698.0,(75)r265.20116.0 3:(0)r259.10694.0,(55)r265.20112.0	1:(32)r259.10700.2,(54)r265.20111.4 2:(14)r259.10696.4,(74)r265.20115.4 3:(53)r265.20111.3,(53)r265.20111.3
11.mpicol.cust.0K10	sing.orig	1:(25)r259.10699.0,(65)r265.20114.0 2:(20)r259.10698.0,(75)r265.20116.0 3:(0)r259.10694.0,(55)r265.20112.0	1:(2)r259.10694.2,(78)r265.20116.3 2:(17)r259.10697.2,(64)r265.20113.4 3:(49)r265.20110.4,(58)r265.20112.3
01.mpicol.0K10	doub.actual	1:(25)r259.10699.0,(65)r265.20114.0 2:(20)r259.10698.0,(75)r265.20116.0 3:(0)r259.10694.0,(55)r265.20112.0	1:(32)r259.10700.2,(54)r265.20111.4 2:(14)r259.10696.4,(74)r265.20115.4 3:(53)r265.20111.3,(53)r265.20111.3

Figure 2: Recommendation Table Sample for one of the bugs

3.5 Concept Lattice and LCA

- Vijay Garg - Applications of lattice theory in distributed systems
- Dimitry Ignatov [?] - Concept Lattice Applications in Information Retrieval
- [15] [18] [3] [17] [33]

3.6 Repetitive Patterns

- [11] [22] [34] [13] [12]

4 RESULTS

Table 1 describes the bug that I injected to ILCS-TSP

4.1 MPI Bugs

5 CASE STUDIES

5.0.1 Case Study: ILCS. ILCS [9]

5.0.2 Case Study: MFEM.

6 DISCUSSION AND FUTURE WORK

REFERENCES

[1] D. H. Ahn, B. R. de Supinski, I. Laguna, G. L. Lee, B. Liblit, B. P. Miller, and M. Schulz. 2009. Scalable temporal order analysis for large scale debugging. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. 1–11. https://doi.org/10.1145/1654059.1654104

[2] Dong H. Ahn, Bronis R. de Supinski, Ignacio Laguna, Gregory L. Lee, Ben Liblit, Barton P. Miller, and Martin Schulz. 2009. Scalable Temporal Order Analysis for Large Scale Debugging. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09)*. ACM, New York, NY, USA, Article 44, 11 pages. https://doi.org/10.1145/1654059.1654104

[3] Michael A. Bender, MartÃŋn Farach-Colton, Giridhar Pemmasani, Steven Skiena, and Pavel Sumazin. 2005. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms* 57, 2 (2005), 75 – 94. https://doi.org/10.1016/j.jalgor.2005.08.001

[4] Ivan Beschastnikh, Jenny Abrahamson, Yuriy Brun, and Michael D. Ernst. 2011. Synoptic: Studying Logged Behavior with Inferred Models. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE '11)*. ACM, New York, NY, USA, 448–451. https://doi.org/10.1145/2025113.2025188

[5] Ivan Beschastnikh, Yuriy Brun, Michael D. Ernst, and Arvind Krishnamurthy. 2014. Inferring Models of Concurrent Systems from Logs of Their Behavior with CSight. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 468–479. https://doi.org/10.1145/2568225.2568246

[6] Ivan Beschastnikh, Yuriy Brun, Michael D. Ernst, Arvind Krishnamurthy, and Thomas E. Anderson. 2011. Mining Temporal Invariants from Partially Ordered Logs. In *Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques (SLAML '11)*. ACM, New York, NY, USA, Article 3, 10 pages. https://doi.org/10.1145/2038633.2038636

[7] Ivan Beschastnikh, Patty Wang, Yuriy Brun, and Michael D. Ernst. 2016. Debugging Distributed Systems. *Commun. ACM* 59, 8 (July 2016), 32–37. https://doi.org/10.1145/2909480

[8] G. Bronevetsky, I. Laguna, S. Bagchi, B. R. de Supinski, D. H. Ahn, and M. Schulz. 2010. AutomaDeD: Automata-based debugging for dissimilar parallel tasks. In *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*. 231–240. https://doi.org/10.1109/DSN.2010.5544927

[9] M. Burtscher and H. Rabeti. 2013. A Scalable Heterogeneous Parallelization Framework for Iterative Local Searches. In *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. 1289–1298. https://doi.org/10.1109/IPDPS.2013.27

[10] Z. Chen, X. Li, J. Chen, H. Zhong, and F. Qin. 2012. SyncChecker: Detecting Synchronization Errors between MPI Applications and Libraries. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*. 342–353. https://doi.org/10.1109/IPDPS.2012.40

[11] M. Crochemore and W. Rytter. 1991. Usefulness of the Karp-Miller-Rosenberg algorithm in parallel computations on strings and arrays. *Theoretical Computer Science* 88, 1 (1991), 59 – 82. http://www.sciencedirect.com/science/article/pii/030439759190073B

[12] Maxime Crochemore and Wojciech Rytter. 1994. *Text Algorithms*. Oxford University Press, Inc., New York, NY, USA.

[13] Maxime Crochemore and Wojciech Rytter. 2002. *Jewels of Stringology*. World Scientific. https://books.google.com/books?id=ipuPQgAACAAJ

[14] Alexander Droste, Michael Kuhn, and Thomas Ludwig. 2015. MPI-checker: Static Analysis for MPI. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC (LLVM '15)*. ACM, New York, NY, USA, Article 3, 10 pages. https://doi.org/10.1145/2833157.2833159

[15] Bernhard Ganter and Rudolf Wille. 1997. *Formal Concept Analysis: Mathematical Foundations* (1st ed.). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[16] Q. Gao, F. Qin, and D. K. Panda. 2007. DMTracker: finding bugs in large-scale parallel programs by detecting anomaly in data movements. In *SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*. 1–12. https://doi.org/10.1145/1362622.1362643

[17] Vijay K. Garg. 2013. Maximal Antichain Lattice Algorithms for Distributed Computations. In *Distributed Computing and Networking*, Davide Frey, Michel Raynal, Saswati Sarkar, Rudrapatna K. Shyamasundar, and Prasun Sinha (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 240–254.

[18] Robert Godin, Rokia Missaoui, and Hassan Alaoui. [n. d.]. INCREMENTAL CONCEPT FORMATION ALGORITHMS BASED ON GALOIS (CONCEPT) LATTICES. *Computational Intelligence* 11, 2 ([n. d.]), 246–267.

[19] Stewart Grant, Hendrik Cech, and Ivan Beschastnikh. 2018. Inferring and Asserting Distributed System Invariants. In *Proceedings of the 40th International Conference on Software Engineering (ICSE '18)*. ACM, New York, NY, USA, 1149–1159. https://doi.org/10.1145/3180155.3180199

[20] K. E. Isaacs, A. Bhatele, J. Lifflander, D. BÃűhme, T. Gamblin, M. Schulz, B. Hamann, and P. Bremer. 2015. Recovering logical structure from Charm++ event traces. In *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12. https://doi.org/10.1145/2807591.2807634

[21] Katherine E. Isaacs, Peer-Timo Bremer, Ilir Jusufi, Todd Gamblin, Abhinav Bhatele, Martin Schulz, and Bernd Hamann. 2014. Combing the Communication Hairball: Visualizing Parallel Execution Traces using Logical Time. *IEEE Transactions on Visualization and Computer Graphics* 20 (2014), 2349–2358.

[22] Richard M. Karp, Raymond E. Miller, and Arnold L. Rosenberg. 1972. Rapid Identification of Repeated Patterns in Strings, Trees and Arrays. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing (STOC '72)*. ACM, New York, NY, USA, 125–136. https://doi.org/10.1145/800152.804905

[23] Andreas Knüpfer, Christian Rössel, Dieter an Mey, Scott Biersdorff, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Daniel Lorenz, Allen D. Malony, Wolfgang E. Nagel, Yury Oleynik, Peter Philippen, Pavel Saviankou, Dirk Schmidl, Sameer Shende, Ronny Tschüter, Michael Wagner, Bert Wesarg, and Felix Wolf. 2011. Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir. In *Tools for High Performance Computing 2011 - Proceedings of the 5th International Workshop on Parallel Tools for High Performance Computing, ZIH, Dresden, September 2011*. 79–91.

[24] Bettina Krammer, Matthias MÃČÅŠller, and Michael Resch. 2004. MPI application development using the analysis tool MARMOT, Vol. 3038. 464–471. https://doi.org/10.1007/978-3-540-24688-6_61

[25] Ignacio Laguna, Dong H. Ahn, Bronis R. de Supinski, Saurabh Bagchi, and Todd Gamblin. 2012. Probabilistic Diagnosis of Performance Faults in Large-scale Parallel Applications. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT '12)*. ACM, New York, NY, USA, 213–222. https://doi.org/10.1145/2370816.2370848

Table 1: Injected Bugs to ILCS-TSP

ID	Level	Bugs	Description
1	MPI	allRed1wrgOp-1-all-x	Different operation (MPI_MAX) in only one (buggyProc) for first MPI_ALLREDUCE() – L229:ilcsTSP.c
2		allRed1wrgSize-1-all-x	Wrong size in only one (buggyProc) for first MPI_ALLREDUCE() – L229:ilcsTSP.c
3		allRed1wrgSize-all-all-x	Wrong Size in all processes for first MPI_ALLREDUCE() – L229:ilcsTSP.c
4		allRed2wrgOp-1-all-x	Different operation (MPI_MAX) in only one (buggyProc) for first MPI_ALLREDUCE() – L277:ilcsTSP.c
5		allRed2wrgSize-1-all-x	Wrong size in only one (buggyProc) for first MPI_ALLREDUCE() – L277:ilcsTSP.c
6		allRed2wrgSize-all-all-x	Wrong Size in all processes for second MPI_ALLREDUCE() – L277:ilcsTSP.c
7		bcastWrgSize-1-all-x	Wrong Size in only one (buggyProc) of MPI_Bcast() – L290:ilcsTSP.c
8		bcastWrgSize-all-all-x	Wrong Size n all processes for MPI_Bcast() – L240:ilcsTSP.c
9	OMP	misCrit-1-1-x	Missing Critical Section in buggyProc and buggyThread – L170:ilcsTSP.c
10		misCrit-all-1-x	Missing Critical Section in buggyThread and all prcoesses – L170:ilcsTSP.c
11		misCrit-1-all-x	Missing Critical Section in buggyProc and all threads – L170:ilcsTSP.c
12		misCrit-all-all-x	Missing Critical Section in all procs and threads – L170:ilcsTSP.c
13		misCrit2-1-1-x	Missing Critical Section in buggyProc and buggyThread – L230:ilcsTSP.c
14		misCrit2-all-1-x	Missing Critical Section in buggyThread – L230:ilcsTSP.c
15		misCrit2-1-all-x	Missing Critical Section in buggyProc and all threads – L230:ilcsTSP.c
16		misCrit2-all-all-x	Missing Critical Section in all procs and threads – L230:ilcsTSP.c
17		misCrit3-1-all-x	Missing Critical Section in buggyProc and all threads – L280:ilcsTSP.c
18		misCrit3-all-all-x	Missing Critical Section in all procs and threads – L280:ilcsTSP.c
19	General	infLoop-1-1-1	Injected an infinite loop after CPU_EXEC() in buggyProc,buggyThread & buggyIter L164:ilcsTSP.c

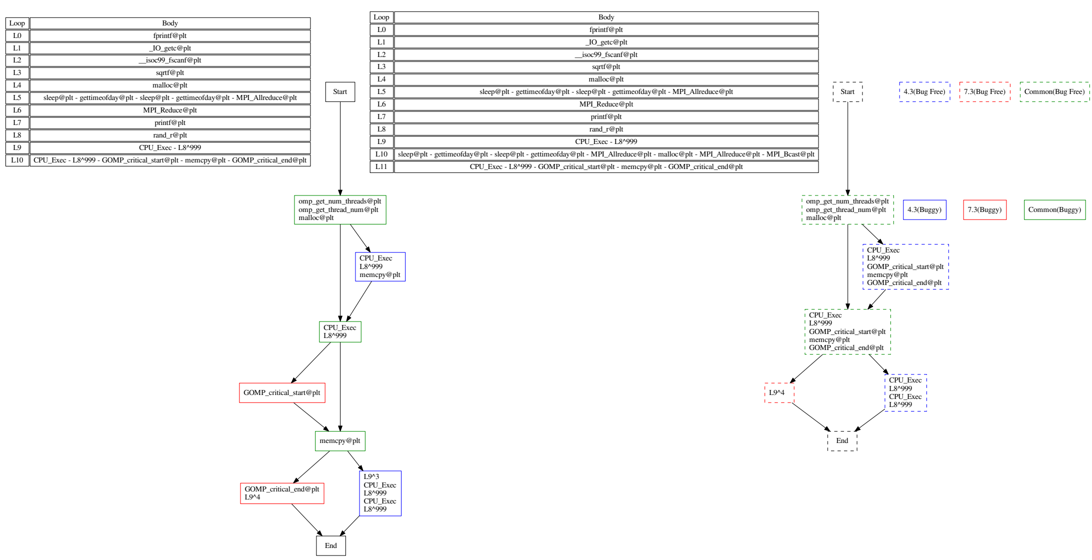


Figure 3: sample diffNLR

[26] Ignacio Laguna, Todd Gamblin, Bronis R. de Supinski, Saurabh Bagchi, Greg Bronevetsky, Dong H. Anh, Martin Schulz, and Barry Rountree. 2011. Large Scale Debugging of Parallel Tasks with AutomaDeD. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. ACM, New York, NY, USA, Article 50, 10 pages. https://doi.org/10.1145/2063384.2063451

[27] Ben Liblit and Alex Aiken. 2002. *Building a Better Backtrace: Techniques for Postmortem Program Analysis.* Technical Report. Berkeley, CA, USA.

[28] Bozhen Liu and Jeff Huang. 2018. D4: Fast Concurrency Debugging with Parallel Differential Analysis. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2018)*. ACM, New York, NY, USA, 359–373. https://doi.org/10.1145/3192366.3192390

[29] Jonathan Mace, Ryan Roelke, and Rodrigo Fonseca. 2018. Pivot Tracing: Dynamic Causal Monitoring for Distributed Systems. *ACM Trans. Comput. Syst.* 35, 4, Article 11 (Dec. 2018), 28 pages. https://doi.org/10.1145/3208104

[30] N. Maruyama and S. Matsuoka. 2008. Model-based fault localization in large-scale computing systems. In *2008 IEEE International Symposium on Parallel and Distributed Processing*. 1–12. https://doi.org/10.1109/IPDPS.2008.4536310

[31] A. V. Mirgorodskiy, N. Maruyama, and B. P. Miller. 2006. Problem Diagnosis in Large-Scale Computing Environments. In *SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*. 11–11. https://doi.org/10.1109/SC.2006.50

[32] Subrata Mitra, Ignacio Laguna, Dong H. Ahn, Saurabh Bagchi, Martin Schulz, and Todd Gamblin. 2014. Accurate Application Progress Analysis for Large-scale Parallel Debugging. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14)*. ACM, New York, NY, USA, 193–203. https://doi.org/10.1145/2594291.2594336

[33] Eugene W. Myers. 1986. An O(ND) Difference Algorithm and Its Variations. *Algorithmica* 1, 2 (1986), 251–266. https://doi.org/10.1007/BF01840446

[34] Atsuyoshi Nakamura, Tomoya Saito, Ichigaku Takigawa, Mineichi Kudo, and Hiroshi Mamitsuka. 2013. Fast algorithms for finding a minimum repetition

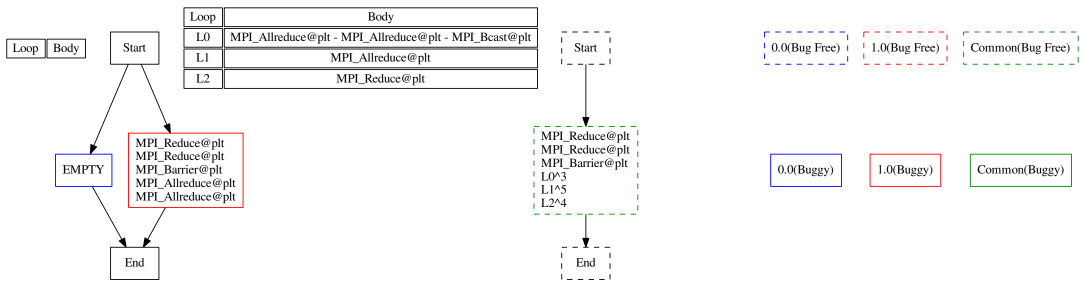


Figure 4: sample diffNLR2

representation of strings and trees. *Discrete Applied Mathematics* 161, 10 (2013), 1556 – 1575. https://doi.org/10.1016/j.dam.2012.12.013

[35] Michael Noeth, Prasun Ratn, Frank Mueller, Martin Schulz, and Bronis R. de Supinski. 2009. ScalaTrace: Scalable compression and replay of communication traces for high-performance computing. *J. Parallel and Distrib. Comput.* 69, 8 (2009), 696 – 710. https://doi.org/10.1016/j.jpdc.2008.09.001 Best Paper Awards: 21st International Parallel and Distributed Processing Symposium (IPDPS 2007).

[36] Robert Preissl, Thomas Köckerbauer, Martin Schulz, Dieter Kranzlmüller, Bronis R. de Supinski, and Daniel J. Quinlan. 2008. Detecting Patterns in MPI Communication Traces. *2008 37th International Conference on Parallel Processing* (2008), 230–237.

[37] Philip C. Roth, Jeremy S. Meredith, and Jeffrey S. Vetter. 2015. Automated Characterization of Parallel Application Communication Patterns. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '15)*. ACM, New York, NY, USA, 73–84. https://doi.org/10.1145/2749246.2749278

[38] Sameer S. Shende and Allen D. Malony. 2006. The Tau Parallel Performance System. *International Journal on High Performance Computer Applications* 20 (May 2006), 287–311. Issue 2. https://doi.org/10.1177/1094342006064482

[39] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. 2002. Automatically Characterizing Large Scale Program Behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*. ACM, New York, NY, USA, 45–57. https://doi.org/10.1145/605397.605403

[40] S. Taheri, S. Devale, G. Gopalakrishnan, and M. Burtscher. [n. d.]. PARLOT: Efficient Whole-Program Call Tracing for HPC Applications. In *7th Workshop on Extreme-Scale Programming Tools, ESPT@SC 2018, Dallas, TX, USA, November 16, 2018*.

[41] M. Weber, R. Brendel, and H. Brunst. 2012. Trace File Comparison with a Hierarchical Sequence Alignment Algorithm. In *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*. 247–254. https://doi.org/10.1109/ISPA.2012.40

[42] Matthias Weber, Ronny Brendel, Tobias Hilbrich, Kathryn Mohror, Martin Schulz, and Holger Brunst. 2016. Structural Clustering: A New Approach to Support Performance Analysis at Scale. IEEE, 484–493. https://doi.org/10.1109/IPDPS.2016.27

[43] Yuan Zhang and Evelyn Duesterwald. 2007. Barrier Matching for Programs with Textually Unaligned Barriers. In *Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '07)*. ACM, New York, NY, USA, 194–204. https://doi.org/10.1145/1229428.1229472