7. CONCLUSION

Existing performance analysis tools and approaches focus on understanding the distribution of runtime costs in an application. However without understanding the value being provided in return for this expense it is difficult to establish where effort is being wasted and where optimisation opportunities might truly exist. In this paper we present a blended analysis approach to quantifying the value provided by all execution paths in an application, thereby enabling an analysis of their efficiency and a practical cost/benefit approach to performance analysis. This allows the discovery of new optimisation opportunities not readily apparent from the original profile data. The results of our experiments and the performance improvements we made in our case studies demonstrate that efficiency analysis is an effective technique that can be used to complement existing performance engineering approaches.

8. ACKNOWLEDGMENTS

David Maplesden is supported by a University of Auckland Doctoral Scholarship.

9. REFERENCES

[1] G. Ammons, T. Ball, and J. R. Larus. Exploiting hardware performance counters with flow and context sensitive profiling. *Proc. of the Conf. on Prog. Language Design and Impl.*, pages 85–96, 1997.

[2] S. Bhattacharya, M. G. Nanda, K. Gopinath, and M. Gupta. Reuse, Recycle to De-bloat Software. *Lecture Notes in Comp. Sci.*, 6813:408–432, 2011.

[3] S. M. Blackburn, R. Garner, C. Hoffmann, A. M. Khan, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, J. E. B. Moss, A. Phansalkar, D. Stefanovic, T. VanDrunen, D. von Dincklage, and B. Wiedermann. The DaCapo Benchmarks: Java Benchmarking Development and Analysis. *Proc. of the Conf. on Object Oriented Prog. Systems Languages and App.*, pages 169–190, 2006.

[4] B. Blanchet. Escape analysis for object-oriented languages: application to java. *Proc. of the Conf. on Object Oriented Prog. Systems Languages and App.*, pages 20–34, 1999.

[5] A. E. Chis, N. Mitchell, E. Schonberg, G. Sevitsky, P. O. Sullivan, T. Parsons, and J. Murphy. Patterns of Memory Inefficiency. *Lecture Notes in Comp. Sci.*, 6813:383–407, 2011.

[6] D. C. D'Elia, C. Demetrescu, and I. Finocchi. Mining hot calling contexts in small space. *Proc. of the Conf. on Prog. Language Design and Impl.*, pages 516–527, 2011.

[7] B. Dufour, B. G. Ryder, and G. Sevitsky. Blended analysis for performance understanding of framework-based applications. *Proc. of the Int'l Symp. on Soft. Testing and Analysis*, pages 118–128, 2007.

[8] B. Dufour, B. G. Ryder, and G. Sevitsky. A scalable technique for characterizing the usage of temporaries in framework-intensive Java applications. *Proc. of the 16th Int'l Symp. on Foundations of Soft. Eng.*, pages 59–70, 2008.

[9] D. Maplesden, E. Tempero, J. Hosking, and J. C. Grundy. Performance Analysis for Object-Oriented Software: A Systematic Mapping. *IEEE Transactions on Soft. Eng.*, 41(7):691–710, 2015.

[10] D. Maplesden, E. Tempero, J. Hosking, and J. C. Grundy. Subsuming Methods: Finding New Optimisation Opportunities in Object-Oriented Software. *6th ACM/SPEC Int'l Conf. on Performance Engineering*, pages 175–186, 2015.

[11] N. Mitchell, E. Schonberg, and G. Sevitsky. Four Trends Leading to Java Runtime Bloat. *IEEE Software*, 27(1):56–63, 2010.

[12] N. Mitchell, G. Sevitsky, and H. Srinivasan. The diary of a datum: an approach to modeling runtime complexity in framework-based applications. *Library-Centric Software Design*, page 85, 2005.

[13] N. Mitchell, G. Sevitsky, and H. Srinivasan. Modeling Runtime Behavior in Framework-Based Applications. *Lecture Notes in Comp. Sci.*, 4067:429–451, 2006.

[14] K. Nguyen and G. Xu. Cachetor: detecting cacheable data to remove bloat. *Proc. of the 9th Joint Meeting on Foundations of Soft. Eng.*, pages 268–278, 2013.

[15] A. Sarimbekov, A. Sewe, W. Binder, P. Moret, and M. Mezini. JP2: Call-site aware calling context profiling for the Java Virtual Machine. *Science of Computer Programming*, 79:146–157, 2014.

[16] A. Sarimbekov, A. Sewe, W. Binder, P. Moret, M. Schoeberl, and M. Mezini. Portable and accurate collection of calling-context-sensitive bytecode metrics for the Java virtual machine. *Proc. of the Int'l Conf. on Principles and Practice of Prog. in Java*, page 11, 2011.

[17] O. Shacham, M. Vechev, and E. Yahav. Chameleon: Adaptive Selection of Collections. *Proc. of the Conf. on Prog. Language Design and Impl.*, pages 408–418, 2009.

[18] G. Xu. Finding reusable data structures. *Proc. of the Conf. on Object Oriented Prog. Systems Languages and App.*, page 1017, 2012.

[19] G. Xu, N. Mitchell, M. Arnold, A. Rountev, E. Schonberg, and G. Sevitsky. Finding low-utility data structures. *Proc. of the Conf. on Prog. Language Design and Impl.*, pages 174–186, 2010.

[20] G. Xu, N. Mitchell, M. Arnold, A. Rountev, E. Schonberg, and G. Sevitsky. Scalable Runtime Bloat Detection Using Abstract Dynamic Slicing. *ACM Transactions Soft. Eng. Methodology*, 23(3):23:1–23:50, 2014.

[21] G. Xu, N. Mitchell, M. Arnold, A. Rountev, and G. Sevitsky. Software Bloat Analysis: Finding, Removing, and Preventing Performance Problems in Modern Large-Scale Object-Oriented Applications. *Proc. of the FSE/SDP Workshop on the Future of Soft. Eng. Research*, pages 421–425, 2010.

[22] G. Xu and A. Rountev. Detecting inefficiently-used containers to avoid bloat. *Proc. of the Conf. on Prog. Language Design and Impl.*, pages 160–173, 2010.

[23] D. Yan, G. Xu, and A. Rountev. Uncovering performance problems in Java applications with reference propagation profiling. *Int'l Conf. on Soft. Eng.*, pages 134–144, 2012.