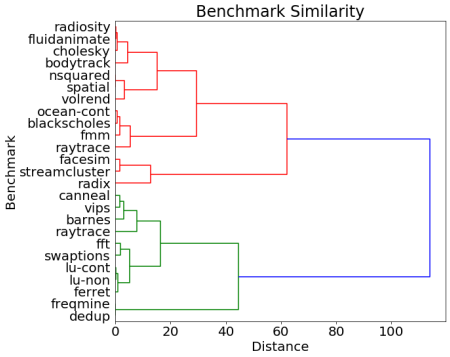
cluster show similar behavior in terms of floating point instructions executed. Using this mechanism one can get a workload set with each workload having similar behavior for the specified bins.

The generated dendrograms can also be used for finding a reduced workload set from a benchmark suite which has a wide coverage of workload behaviors across the entire suite. Instead of selecting workloads having similar behavior, we can select workloads with diverse behaviors to increase coverage. The methodology for finding a reduced set from a dendrogram is described in [14].

Figure 4: Dendrogram between PARSEC and SPLASH



4 FUTURE WORK

4.1 Extending FAB

FAB is easily extensible owing to pre-defined hierarchy of the underlying database of workload characteristics. In the future, we plan on adding more analytical and visualization features to the front-end, as well as more information to the back-end to enable it to carry out multiple types of analyses. Some of the proposed future work envisioned for FAB is enumerated below.

- Enable FAB analyses to support a richer set of metrics like working set size and branch transition rates, in addition to instruction mixes.
- Enable FAB to analyse a broader set of benchmark suites by adding characterization data from suites like BioPerf (bioinformatics), MediaBench (media) and Moby (mobile applications).
- Enable FAB to carry out cross-ISA analysis of workloads by adding characterization data for suites across multiple ISAs like ARM and RISC-V.

4.2 Workload Creation

We also aim to augment the functionality of FAB to support workload creation by generating synthetic workloads with user specified characteristics. Previous work has shown that statistical profiles of workloads can be used for generating synthetic workloads [9]. These workloads are typically shorter than the actual workloads, leading to reduced simulation time. By using these techniques, the performance characteristics can quickly converge to a steady state which makes such workloads suitable for accelerated design space exploration[8]. Enhancements to specific components of an architecture can be tested by generating workloads that heavily utilize that component.

In future, we aim to use instruction mixes to train statistical machine learning models like generative adversarial networks to generate synthetic workloads using the framework. We envision that the computer architect will be able to define not only the composition of the synthetic workload using bins but also specify different phases that she is interested in observing. Not only that, an architect might be interested in observing the behavior of a proposed architectures under a sequence of pre-specified program phases, which might be harder to find in existing applications. For example, the architect might want to observe the proposed architecture under a integer compute-intensive phase, followed by a memory-intensive phase, followed by a floating point compute-intensive phase. Allowing for the creating of such user-specified workloads, which still have a basis in "real" workloads will help reason about system design and help architects explore what-if scenarios.

5 CONCLUSIONS

In this paper, we introduce FAB - an extensible and flexible framework that allows computer architects to analyze and compare characteristics of workloads within one, and across multiple benchmark suites.

The framework currently can be used for analyzing and selecting workloads that fulfill computer architect specified criteria. By providing visual aids like dendograms, the framework allows for selecting workload sets across multiple benchmark suites by allowing for removal of workloads with similar behavior. We aim to extend FAB to support multiple ISAs and provide richer set of options to the user for specifying workload characteristics.

ACKNOWLEDGMENTS

This work was supported in part by SERB grant ECR/2017/000887, IIT Gandhinagar and Ashoka University.

REFERENCES

[1] [n. d.]. OpenBLAS : An optimized BLAS library. http://www.openblas.net/
[2] [n. d.]. SPEC CPU 2017 Documentation. https://www.spec.org/cpu2017/
[3] 2016. Intel 64 and IA-32 Architectures Software Developers Manual Volume 2A: Instruction Set Reference, A-L.
[4] D. H. Bailey et al. 1991. The NAS Parallel Benchmarks&Mdash;Summary and Preliminary Results. In *Proceedings of Supercomputing*.
[5] Christian Bienia et al. 2008. *The PARSEC Benchmark Suite: Characterization and Architectural Implications.* Technical Report TR-811-08. Princeton University.
[6] C. Bienia et al. 2008. PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on Chip-Multiprocessors. In *Proceedings of IISWC*.
[7] Jack J. Dongarra et al. 2002. The LINPACK benchmark: Past, present, and future.
[8] Lieven Eeckhout. 2010. *Computer Architecture Performance Evaluation Methods* (1st ed.). Morgan & Claypool Publishers.
[9] L. Eeckhout et al. 2000. Performance analysis through synthetic trace generation. In *Proceedings of ISPASS*.
[10] C. Sakalis et al. 2016. Splash-3: A properly synchronized benchmark suite for contemporary research.
[11] Chi-Keung Luk et al. 2005. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation. In *Proceedings PLDI*.
[12] John D. McCalpin. 1995. Memory Bandwidth and Machine Balance in Current High Performance Computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter* (Dec. 1995), 19–25.
[13] D. A. Menasce. 2002. TPC-W: a benchmark for e-commerce. *IEEE Internet Computing* 6, 3 (2002).
[14] R. Panda et al. 2018. Wait of a Decade: Did SPEC CPU 2017 Broaden the Performance Horizon?. In *Proceedings of HPCA*.
[15] Timothy Sherwood et al. 2002. Automatically Characterizing Large Scale Program Behavior. *SIGOPS Oper. Syst. Rev.* 36, 5 (2002).