Similarity Measurement for Proxy Application Fidelity

Extended Abstract

Si Chen
Emory University
Atlanta, Georgia
si.chen2@emory.edu

Omar Aaziz
Sandia National Laboratories
Albuquerque, New Mexico
oaaziz@sandia.gov

Jeanine Cook
Sandia National Laboratories
Albuquerque, New Mexico
jeacook@sandia.gov

Avani Wildani
Emory University
Atlanta, Georgia
avani@mathcs.emory.edu

ABSTRACT

Proxy applications, designed to represent similar but much larger and more complex parent applications, are widely used for system co-design and procurement. However, quantitative validation is missing to prove that proxies are faithful representations of their parents. In this work, we apply several machine learning-based methods to quantitatively demonstrate the correspondence of proxy to parent in their underlying node and memory hardware behavior. We show that different similarity measurements have similar results while evaluating the proxy application fidelity, and most proxies are good representations of their parents.

KEYWORDS

Co-design, proxy/parent representativeness, similarity measurement

1 INTRODUCTION

Co-design is crucial for system designs to achieve optimal application performance. While some science applications have thousands of lines of code and complex dependencies, proxy applications are open-source, smaller, more tractable models which represent some characteristics of their parent applications. Many primary computer system vendors use proxy applications for performance prediction in activities like system co-design and procurement. It is thus critical to ensure that the proxy applications used are high fidelity models of target applications. Specifically, our focus is to quantitatively demonstrate proxy fidelity.

2 METHODOLOGY

We collect data and use that data to calculate similarity along three different statistical metrics: cosine similarity, Jensen-Shannon divergence, and the Wasserstein distance. We use two system platforms with different architectures (Intel Skylake and IBM Power 9) to collect data on 21 applications that span several scientific domains. We collected PAPI [4] hardware performance counters using LDMS [2] to give us a large array of measurements to understand how similar applications utilize system resources in term of node computation and memory.

To make the performance comparable, we run each application with the same input problem and parameters where possible. Data is collected from each application process (*i.e.,* each MPI rank), and we compute an average for each event. In the end, every application on each platform is represented by a vector with 700 dimensions (features), each of which is an average rank value for one hardware event. Moreover, we create several architectural subgroups (selected by experts) for these features, including cache, branch prediction, pipeline, instruction mix, memory, and virtual memory. We then use our distance metrics to evaluate similarity for overall vectors as well as subset vectors.

Cosine similarity, $\cos(\theta) = \frac{\sum_{i=1}^{n} x_i y_i}{\|x\|\|y\|}$, compares the angle between two vectors x and y using vector inner product. Here, while 0 degree and 90 indicate similar (the same) and dissimilar respectively.

Jensen-Shannon (JS) divergence, on the other hand, measures the distance between two probability distributions, P and Q. JS divergence is defined as $\mathrm{JSD}(P\|Q) = \frac{1}{2}D(P\|M) + \frac{1}{2}D(Q\|M)$, which calculates the total divergence to the average. Unlike the well-known Kullback–Leibler metric, JS divergence is symmetric and returns a value between 0 and 1, both of which are valuable properties for application comparison.

Wasserstein distance, also known as earth mover distance, is defined as $W_2(p,q) = \sqrt{\min_{P_{XY}} E_{P_{XY}}\left[\|x-y\|_2^2\right]}$ s.t. $P_X \sim p, P_Y \sim q$. Wasserstein distance can be interpreted as the minimum "cost" of transforming a probability distribution X into distribution Y.

After calculating these three similarity metrics, we use hierarchical clustering to further rank the applications. We consider applications at the same branch level of the cluster hierarchy to be more similar than applications at different depths.

3 RESULT

We test 21 applications from the Exascale Computing Project [1]. Most applications are members of proxy/parent pairs: ExaMiniMD/LAMMPS, miniQMC/QMCPACK, miniVite/Vite, Nekbone/Nek5000, PICSARlite/PICSAR, SW4lite/SW4, SWFFT/HACC, and XSBench/OpenMC. We also investigate five extra proxy applications that lack designated parents to evaluate our methodology. These applications are: AMG2013, Laghos, PENNANT, SNAP, and Castro.

We set up the experiment on two system platforms with different architectures, Intel Skylake (sky) and IBM Power9 (p9). We run all of our applications in MPI-only mode, using 128 ranks (one rank per core), on four nodes. We only run each experiment once. We are fetching more data for future work.

Figure 1, Figure 2, and Figure 3 are the overall feature results of three distance methods in the Skylake system. The spectrum of

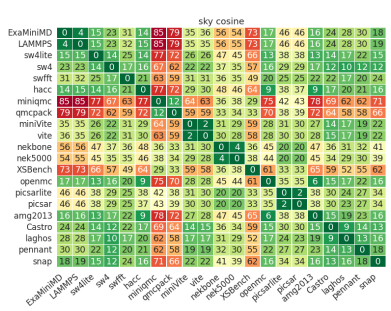

Figure 1: Cosine Similarity, Skylake

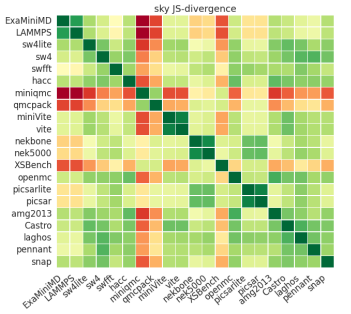

Figure 2: JS distance, Skylake

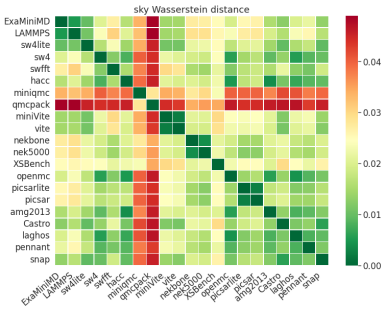

Figure 3: Wasserstein distance, Skylake

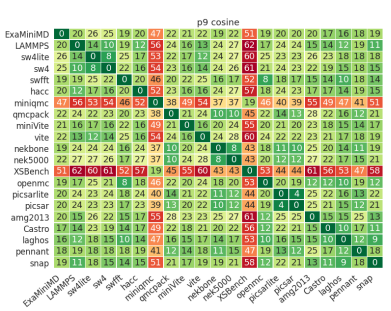

Figure 4: Cosine Similarity, Power9

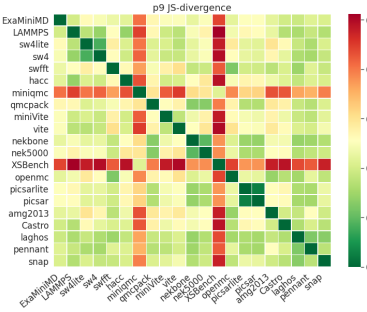

Figure 5: JS distance, Power9

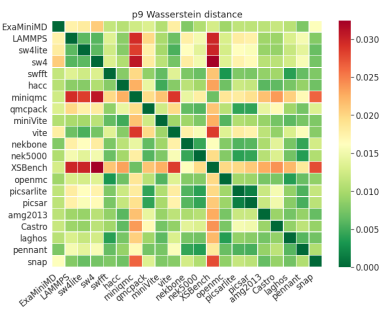

Figure 6: Wasserstein distance, Power9

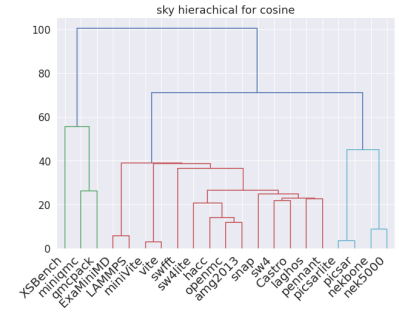

Figure 7: Hierarchical Clustering for cosine similarity, Skylake

colors represents similarity, where dark green is highly similar and dark red is highly dissimilar. We can see that, generally, the three distance metrics we evaluate return similar correlations. Two applications (QMCPack and MiniQMC) are similar but very different from other applications. miniVite/vite, PICSARlite/PICSOR, and Nek5000/Nekbone, are the most similar proxy/parent application pairs. Of these, Nek5000/Nekbone and vite/miniVite show very high cosine similarity, and all three have relatively high JS and Wasserstein. The five unpaired proxy applications used as controls show relative similarity to each other, but do not match the known pairs.

In our hierarchical clustering (Figure 7), we observe that the three most similar proxy/parent pairs exhibit the shortest branch heights, and furthermore that the overall application similarity tracks that of our 3 core distance metrics. Unsurprisingly, our results vary across platforms because of the differences in hardware characteristics (Figure 4, 5, 6). Overall and subset results do not have much difference, and three subgroups (Cache-related, Instruction mix, and Pipline) contribute most to the overall similarity matrix.

4 RELATED WORK

Proxy application characterization, including comparison of individual proxies to their parent applications and performance projection, is well studied [3; 6]). Our work serves as a metaananalysis across similarity methods, demonstrating their interchangeability in this domain. Other cosine similarity metrics include Square Root Cosine Similarity and Improved Square Root Cosine Similarity [5], and we plan to incorporate these in a future analysis.

5 CONCLUSION

We have explored three similarity metrics to evaluate how proxy and parent applications relate to each other and to other applications. We show that most proxies are good representations of their parents, with minor divergences, independent of the similarity metric used. Since these distance methods show similar results, we pose that cosine similarity should become the preferred similarity metric because of its simplicity, performance, and ease of interpretability by geometric angle. For the next step, we will calculate self-similarity for time series data with sliding windows to further validate the similarity methods as well as gather performance numbers for cosine similarity. Additionally, we are working to validate parent/proxy similarity through measuring the performance discrepancy when parent applications are run on systems designed for the proxy applications meant to represent them compared to native systems.

REFERENCES

[1] Exascale computing project. https://proxyapps.exascaleproject.org/app/.

[2] Anthony Agelastos et al. The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications. In *SC'14: Proc. International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 154–165, Piscataway, NJ, USA, 2014. IEEE Press.

[3] Richard F Barrett, Paul S Crozier, DW Doerfler, Michael A Heroux, Paul T Lin, HK Thornquist, TG Trucano, and Courtenay T Vaughan. Assessing the role of mini-applications in predicting key performance characteristics of scientific and engineering applications. *Journal of Parallel and Distributed Computing*, 75:107–122, 2015.

[4] Philip J Mucci, Shirley Browne, Christine Deane, and George Ho. PAPI: A Portable Interface to Hardware Performance Counters. In *Proceedings of the department of defense HPCMP users group conference*, volume 710, 1999.

[5] Sahar Sohangir and Dingding Wang. Improved sqrt-cosine similarity measurement. *Journal of Big Data*, 4(1):1–13, 2017.

[6] Miwako Tsuji, William TC Kramer, and Mitsuhisa Sato. A performance projection of mini-applications onto benchmarks toward the performance projection of real-applications. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 826–833. IEEE, 2017.