SimPoint++: Less Simulation Points

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

Simon Garcia De Gonzalo
Sandia National Laboratories
Albuquerque, New Mexico
simgarc@sandia.gov

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

***Abstract*—SimPoint has a wide-ranging impact on computer architecture research for automatically finding a small set of simulation points to represent the complete execution of a program for efficient and accurate simulations. While many studies have used SimPoint as part of their methodology, there's been little consideration of whether the set of Simulation Points that Simpoint provides is as small as possible. We propose SimPoint++, which replaces the BIC method by combining WCSS and Silhouette to find the optimal cluster number. The new Python framework of SimPoint++ also provides a dimension reduction pipeline for effective clustering and supports multi-thread application analysis. We evaluate SimPoint++ with SPEC CPU 2017 benchmarks. SimPoint++ achieves comparable or higher accuracy with significantly fewer simulation points, resulting in a 5x speed-up in simulation time compared to state-of-the-art solutions.**

***Index Terms*—checkpoint, simulation, clustering, sampling**

I. Introduction

SimPoint [12] is a widely adopted technique for simulation acceleration in computer architecture. It uses statistical sampling and clustering to capture representative program execution regions. SimPoint has been integrated into numerous tools (*e.g.*, PinPoint [8], BarrierPoint [2], and LoopPoint [10]) and simulation frameworks (*e.g.*, Gem5 [7] and Sniper [1]), revolutionizing microprocessor simulation and performance analysis. However, the most recent version (SimPoint 3.0) from 2006 lacks advanced dimension reduction and clustering algorithms. Furthermore, SimPoint often identifies more representative points than necessary, increasing simulation time.

We propose **SimPoint++**, an improved version that addresses SimPoint's limitations, particularly in determining the optimal cluster number K. **SimPoint++** employs the within-cluster sum of squares (WCSS) and silhouette score to estimate and fine-tune the range of K values, replacing the less effective Bayesian Information Criterion (BIC) method. This approach achieves comparable accuracy with a significantly smaller optimal K, in both serial and multi-thread benchmarks, potentially reducing simulation time while maintaining accuracy. By integrating advanced algorithms and providing a more robust approach for determining optimal clusters, **SimPoint++** updates SimPoint with new techniques in data analysis.

II. Background

A. Original SimPoint Workflow

SimPoint is based on the observation that programs often exhibit repetitive behavioral patterns over time, with program behavior directly linked to the code executed during specific intervals. When intervals show similar code patterns or "fingerprints", they typically have comparable performance characteristics and can be represented by a single sample. By identifying these representative patterns and selecting one sample point from each, SimPoint enables rapid sampling that accurately reconstructs the program's entire execution process, significantly reducing simulation time while maintaining high accuracy in performance analysis.

The SimPoint process involves several steps: First, the program is sliced into chunks, and Basic Block Vectors (BBVs) are generated for each chunk to summarize its behavior. Random Projection is then applied to the BBVs to reduce dimensionality and limit computational complexity. Next, the K-means algorithm clusters the reduced BBVs, grouping similar program regions. From each cluster, a single region is selected as a representative Sim_Point, effectively encapsulating the program's behavior. Finally, each Sim_Point is assigned a weight representing the relative frequency of its corresponding behavior in the overall program execution.

B. Why do we need to replace BIC in SimPoint?

The Bayesian Information Criterion (BIC) is a statistical measure for model selection, balancing model likelihood with complexity. SimPoint employs BIC to determine the optimal cluster number K. However, the SimPoint authors identified a limitation in that BIC scores tend to increase with the number of clusters, which could lead to selecting the maximum possible K. Thus, instead of choosing the maximum BIC score, they use binary search to select a clustering that achieves 90% of the BIC score range, identifying a point beyond which the score increases only marginally. The problem of determining the best K is then transferred to choosing an appropriate MAX K, with complex recommendations. However, the 0.9 threshold, while shown to minimize IPC variance within clusters, may not be universally optimal across all programs or architectures. Furthermore, the method's dependency on MAX K may not always reflect the true underlying structure of the data, and it could potentially overlook more optimal solutions beyond the chosen MAX K. To select simulation points representing the top percent of execution, SimPoint 3.0 offers a Coverage option, allowing users to select only the largest clusters that constitute the majority of the program's weight. While this strategy could reduce the number of clusters, it is ineffective when cluster sizes are similar.

Despite all these efforts, they do not fully resolve the underlying issues. The fundamental problem lies in BIC's assumption of a Gaussian distribution, which is often not applicable to program behavior data. Program behavior distributions are diverse, with varying characteristics across clusters and features. This deviation from the Gaussian assumption can lead

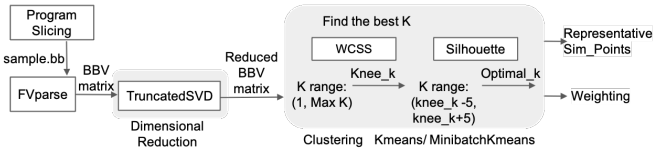

Fig. 1: Workflow of **SimPoint++**

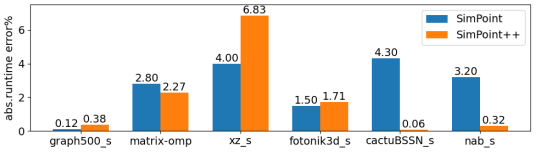

Fig. 2: Absolute prediction error: SimPoint VS **SimPoint++**

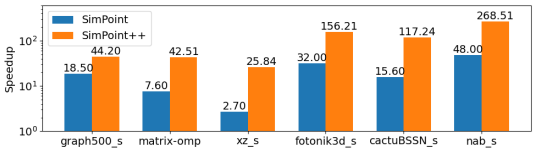

Fig. 3: Speed up: SimPoint VS **SimPoint++**

to misinterpretation of cluster characteristics and makes BIC difficult to use as a reliable criterion for program behavior problems. Therefore, we need a more effective method for determining the optimal K that doesn't rely on potentially arbitrary thresholds or MAX K values.

III. Methodology

SimPoint++ inherits the architectural principles of SimPoint. Two updated components are highlighted with a gray background in the **SimPoint++** workflow (Fig. 1). Visit https://github.com/meditates/Simpoint++.git for details.

A. Dimension Reduction

While SimPoint uses Random Projection [3] for computational efficiency and ease of implementation, the default reduced dimension number of 15 may not be sufficient to preserve information for complex application behaviors, according to the Johnson-Lindenstrauss lemma [6]. **SimPoint++** employs Truncated SVD, aka Latent Semantic Analysis LSA [5], which outperforms Principal Component Analysis (PCA) in speed and efficiency for high-dimensional, sparse data.

B. Clustering

a) Updated K-means: Instead of the vanilla K-means used in SimPoint, **SimPoint++** adopts the k-means++ initialization scheme [9], selecting initial cluster centroids using sampling based on an empirical probability distribution of the points' contributions to the overall inertia. This technique speeds up convergence without multiple runs for each K. When dealing with a huge dataset, **SimPoint++** employs MiniBatchKMeans [11], a variant of K-means that processes data in mini-batches to reduce computation time.

b) Find the best K: Determining the optimal number of clusters K is crucial in K-means clustering, and combining different methods can yield a more reliable result. **SimPoint++** uses both WCSS and Silhouette methods. WCSS measures the sum of squared distances between data points and their assigned cluster centers, aiming to minimize this value to ensure tight clusters. The "elbow" point on the WCSS curve indicates where the benefit of adding more clusters diminishes. The Silhouette score evaluates how well each point matches its assigned cluster versus neighboring clusters, with higher scores indicating better clustering. **SimPoint++** first applies the KneeLocator method on the WCSS curve to identify the elbow point `knee_k`, which helps narrow down the search range for K to (`knee_k` - 5, `knee_k` + 5). Then, the Silhouette scores are calculated within this range, and the number of clusters with the highest score is selected as the `optimal_k`. This combination of WCSS and Silhouette provides a robust approach for determining the optimal number of clusters, especially in complex or noisy datasets where the WCSS elbow point might be unclear.

IV. Experiment Result

We conducted experiments on Cloudlab [4] using an x86 architecture node (c220g2). Our tests included a serial application, Graph500, and a demo multi-threaded application, Matrix-OMP. Four other applications from SPEC CPU 2017 are multi-threaded with train input. The serial application was simulated using Gem5, while the multi-threaded applications were simulated using the LoopPoint [10] platform on the Sniper simulator, with passive wait policies and 8 threads.

SimPoint++ demonstrates superior performance compared to SimPoint, achieving lower error rates (Fig. 2) and higher speed-ups compared to full simulation (Fig. 3). The absolute prediction error for **SimPoint++** is 1.93%, which is lower than the 2.65% error observed with SimPoint. Furthermore, the average speed-up for **SimPoint++** is 109, more than five times the 20.7 speed-up achieved with SimPoint.

To our knowledge, no articles have specifically addressed improving SimPoint's performance. This may be due to SimPoint's established effectiveness, leading researchers to use it as-is. Additionally, SimPoint's C++ codebase, with algorithms implemented from scratch, complicates the integration of advanced techniques.

V. Conclusion

SimPoint++ offers enhanced techniques for dimension reduction and determining the optimal number of clusters, resulting in significantly fewer clusters than SimPoint. We are currently exploring additional datasets and alternative clustering methods (*e.g.*, spectral clustering) to further improve **SimPoint++**. The potential reduction in simulation time, combined with comparable accuracy, could greatly accelerate computer architecture research and development, enabling faster iterations in processor and application design and optimization.

VI. Acknowledgements

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

[1] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–12.

[2] T. E. Carlson, W. Heirman, K. Van Craeynest, and L. Eeckhout, "Barrierpoint: Sampled simulation of multi-threaded applications," in *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2014, pp. 2–12.

[3] S. Dasgupta, "Experiments with random projection," *arXiv preprint arXiv:1301.3849*, 2013.

[4] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb *et al.*, "The design and operation of cloudlab." in *USENIX Annual Technical Conference*, 2019, pp. 1–14.

[5] Ioana, "Latent Semantic Analysis: intuition, math, implementation," https://towardsdatascience.com/latent-semantic-analysis-intuition-math-implementation-a194aff870f8, may 2020.

[6] W. B. Johnson, J. Lindenstrauss, and G. Schechtman, "Extensions of lipschitz maps into banach spaces," *Israel Journal of Mathematics*, vol. 54, no. 2, pp. 129–138, 1986.

[7] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, B. Beckmann, S. Bharadwaj *et al.*, "The gem5 simulator: Version 20.0+," *arXiv preprint arXiv:2007.03152*, 2020.

[8] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi, "Pinpointing representative portions of large intel® itanium® programs with dynamic instrumentation," in *37th International Symposium on Microarchitecture (MICRO-37'04)*. IEEE, 2004, pp. 81–92.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] A. Sabu, H. Patil, W. Heirman, and T. E. Carlson, "Looppoint: Checkpoint-driven sampled simulation for multi-threaded applications," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 604–618.

[11] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1177–1178.

[12] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," *ACM SIGPLAN Notices*, vol. 37, no. 10, pp. 45–57, 2002.