Approximating with Input Level Granularity

Parker Hill, Michael A. Laurenzano, Mehrzad Samadi, Scott Mahlke, Jason Mars, Lingjia Tang
University of Michigan - Ann Arbor, MI
{parkerhh, mlaurenz, mehrzads, mahlke, profmars, lingjia}@umich.edu

Abstract

Approximate computing is a technique for bridging the growing imbalance between computational power and computational needs by trading small amounts of accuracy for large amounts of performance or energy. Current approximate computing techniques are configured to choose how to approximate based either on training inputs that are representative of the "real" input or on occasional runtime checks that compare the exact results to those of the approximation and adjust accordingly. We argue that because these approaches are based on worst or average case behavior, they cannot make the most of each input and thus they are bound to either cause excessive error or leave performance on the table. We introduce input level granularity, an approach that may make it possible to achieve good performance and acceptable accuracy on every input.

1. Background

A number of approximate computing systems determine the approximation method offline [1–3]. This requires that a representative training input set is available to determine whether or not an approximation technique is acceptable. A representative training input set must consist of inputs that accurately model the distribution of possible inputs that may arise during execution. Although it is possible, with domain specific knowledge, to build ideal training sets, the characteristics of such systems are still restricted to choose between unacceptable accuracy or conservative performance before execution.

Another technique that has been proposed involves periodically calibrating the system by computing the approximate and exact outputs for the current input [4, 5]. During this calibration step, if the error is too high, then the system will resort to a less aggressive approximation method. Similarly, it will become more aggressive if the error is below a specific accuracy threshold. Although this has more flexibility than the static approach, the frequency parameter of periodic calibration forces the user of this method to make a similar decision between meeting the target output quality and providing high performance.

Based on the drawbacks of these techniques, we believe that an important next step in approximate computing is to build systems that dynamically choose the approximation method based only on the current input being processed. This circumvents the issue of picking a training set, since each input is treated independently. Additionally, when all choices are based only on the current input, the issue between violating accuracy constraints and being overly conservative is nonexistent.

2. Input Level Granularity

When difficult to approximate, but unlikely, inputs occur, a static system will consistently yield unacceptable results for these inputs. Although on average the static system will appear to perform correctly, it is not unlikely that there exists a correlation between error and specific inputs which may provide a poor user experience. For example, if an intelligent personal assistant always approximates the audio from a very weak signal

as it would for a strong signal, then the specific users in this situation will continuously receive unusable results. On the other hand, it would not be practical to select an approximation configuration that would satisfy all situations. This level of approximation would be unlikely to provide any substantial performance gains at all.

In order to provide more aggressive approximation methods while still maintaining acceptable accuracy per input, rather than on average, we propose a system that makes decisions at an input level granularity. This is also favorable from the user's point of view, since the user expects that each result that they see meets a certain quality threshold.

A high level illustration of how such a system might work is provided in Figure 1. First, the input is given to an analysis mechanism that observes various characteristics of the input. Using these parameters, the analysis engine marks each approximation method with an expected accuracy level and speedup. The expected runtime properties are given to the selection module. This module picks the approximation method with the highest performance that is expected to meet the accuracy bound set by the user. For example, Method1 in the figure may be excluded if the desired accuracy level is 90% and Method2 is selected over Method3, since Method2 has higher speedup. Finally, the full input is passed to the selected approximation algorithm to produce the output.

Assuming that the analysis provided representative metrics, then the chosen approximation will be approaching maximal performance across contrasting inputs without increasing the risk of violating an accuracy target. The two sources of reduced performance in the system are from incorrectly analyzing an approximation method and from the overhead of the analysis.

The central challenge in building this system is that the analysis engine must be engineered in such a way that it provides reasonably accurate error and performance estimates of each approximation method, but at the same time it must execute quickly to avoid diminishing the gains of approximating. The optimal set of input or output values to observe and the right characteristics of these sets to use is critical to the success of this technique.

Although it is not trivial to determine an appropriate mechanism for analysis, we believe that the intrinsically low maximum performance gains of static and

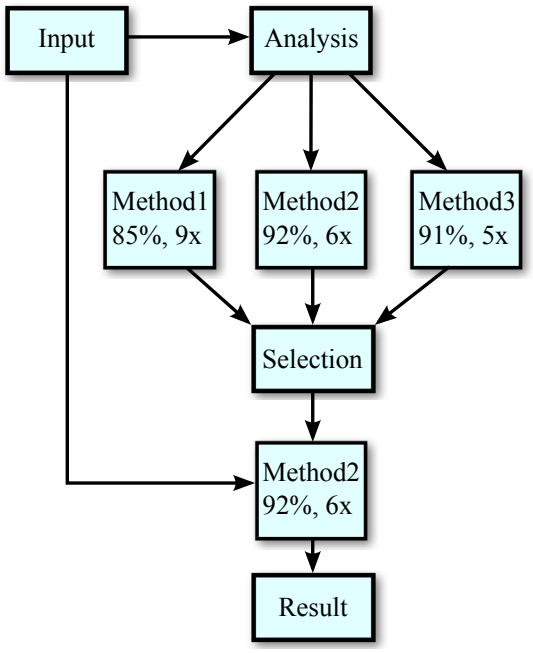

Figure 1: High level design of an approximation runtime system that makes choices at an input level granularity.

calibration based systems will result in the success of input dependent approximation techniques. Static approximation is limited by the most difficult to approximate case and calibration-based approximation is limited because it occasionally calculates exact results. Approximation with input level granularity, on the other hand, is only limited by the amount of time required to calculate reasonable estimates of the characteristics of the input.

References

[1] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. Neural acceleration for general-purpose approximate programs. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 449–460. IEEE Computer Society, 2012.

[2] H. Hoffmann, S. Misailovic, S. Sidiroglou, A. Agarwal, and M. Rinard. Using code perforation to improve performance, reduce energy consumption, and respond to failures. In *MIT Tech Report (MIT-CSAIL-TR-2009-042)*, 2009.

[3] M. Rinard. Probabilistic accuracy bounds for fault-tolerant computations that discard tasks. In *International Conference on Supercomputing (ICS)*, 2006.

[4] M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke. Paraprox: Pattern-based approximation for data parallel ap-

plications. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*, pages 35–50. ACM, 2014.

[5] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke. SAGE: Self-tuning approximation for graphics engines. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 13–24. ACM, 2013.