Architecture-Aware Approximate Computing

Mustafa Karakoy
TOBB University of Economics and Technology
Ankara, TURKEY
m.karakoy@yahoo.co.uk

Orhan Kislal
Pennsylvania State University
State College, PA, USA
kislal.orhan@gmail.com

Xulong Tang
Pennsylvania State University
State College, PA, USA
xzt102@psu.edu

Mahmut Taylan Kandemir
Pennsylvania State University
State College, PA, USA
mtk2@psu.edu

Meenakshi Arunachalam
Intel
Hillsboro, Oregon, USA
meena.arunachalam@intel.com

ABSTRACT

Observing that many application programs from different domains can live with less-than-perfect accuracy, existing techniques try to trade off program output accuracy with performance-energy savings. While these works provide point solutions, they leave three critical questions regarding approximate computing unanswered: (i) what is the maximum potential of skipping (i.e., not performing) data accesses under a given inaccuracy bound?; (ii) can we identify the data accesses to drop randomly, or is being architecture aware critical?; and (iii) do two executions that skip the same number of data accesses always result in the same output quality (error)? This paper first provides answers to these questions using ten multithreaded workloads, and then presents a program slicing-based approach that identifies the set of data accesses to drop. Results indicate 8.8% performance improvement and 13.7% energy saving are possible when we set the error bound to 2%, and the corresponding improvements jump to 15% and 25%, respectively, when the error bound is raised to 4%.

CCS CONCEPTS

• **Computer systems organization → Multicore architectures**;

KEYWORDS

Approximate computing; compiler; manycore system

ACM Reference Format:
Mustafa Karakoy, Orhan Kislal, Xulong Tang, Mahmut Taylan Kandemir, and Meenakshi Arunachalam. 2019. Architecture-Aware Approximate Computing. In *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '19 Abstracts), June 24–28, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3309697.3331508

1 INTRODUCTION

Constraints imposed by hardware scaling and data and control dependencies across computations combined by compilers' inability to maximize computation parallelism limit the performance potential of multithreaded workloads running on emerging manycore systems. State-of-the-art research on computer architecture [2], optimizing/parallelizing compilers [4, 8, 9] and runtime systems/OS [5, 6] helps us extract increasingly more performance from modern architectures, but their impact is hampered by ever-growing application and hardware complexities.

Many workloads in different application domains can live with a "less-than-perfect" output quality. The application programmers are usually provided with metrics to evaluate the output quality of a particular application. In this paper, we target the application domains where the programmers have the capability to determine the error bound of application outputs. Observing this, performance and energy benefits that arise from deliberate use of so-called "approximate computing" has recently been explored across software and hardware stacks [1, 3, 7, 10]. While prior art [5] in approximate computing focused on point solutions that typically trade off accuracy (output quality) with performance and energy benefits, existing works do not target evaluating the maximum potential of approximate computing or proposing practical schemes that can come close to this potential. In particular, we believe that the prior research leaves three critical questions regarding approximate computing unanswered, especially in the context of dropping/skipping costly data (cache/memory) accesses in data intensive applications:

- What is the maximum potential of skipping (i.e., not performing) data accesses under a given inaccuracy (output quality) bound?
- Can we simply identify the data accesses to drop randomly, or is being architecture aware (i.e., identifying the "costliest" data accesses with respect to a given architecture) critical?
- Do two executions that skip the same number of data accesses always result in the same output quality (error)?

Motivated by these questions, we make two main **contributions** in this work:

- First, we explore the potential benefits of a form of approximate computing that drops select data accesses during the execution of parallel workloads on emerging network-on-chip (NoC) based manycore systems. The unique aspect of this evaluation is its "architecture awareness". That is, given a bound on inaccuracy (the minimum level of program output quality that can be tolerated by user/execution environment), we quantify the benefits of dropping the "costliest" data accesses (in our manycore architecture), as opposed to dropping data accesses "randomly". Our

SIGMETRICS '19 Abstracts, June 24–28, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6678-6/19/06.
https://doi.org/10.1145/3309697.3331508

experiments with ten different multithreaded workloads indicate that *being architecture aware* in dropping data accesses pays off, resulting in 27% additional performance improvement, over randomly dropping the same number of data accesses. Unfortunately, our results also indicate that two different executions of a given application that drop the same number of data accesses can result in quite different output quality values (errors), which makes it difficult to maximize performance under a given error bound.

- Second, motivated by this last observation above, we propose a "program slicing" based approach that identifies the set of data accesses to drop such that we (i) maximize the resulting performance/energy benefits and (ii) remain within the error (inaccuracy) bound specified by the user. Our slicing based approach first uses backward slicing and then forward slicing to decide the set of data accesses to drop. Our experimental evaluations of this slicing-based approach under multithreaded workloads and a cycle-accurate manycore simulator show that, when averaged over all ten benchmark programs we have, 8.8% performance improvement and 13.7% energy saving are possible when we set the error bound to 2%, and the corresponding improvements jump to 15% and 25%, respectively, when the error bound is set to 4%. We also tested a restricted version of our approach on a commercial manycore system, and observed 6.8% and 11.2% average performance improvements with error bounds of 2% and 4%, respectively.

2 DROPPING DATA ACCESSES

While there exist different approaches to approximate computing, each leading to a different tradeoff between performance/energy benefits and program output accuracy, in this work we use "skipping/dropping data accesses". Three other possible approaches are (i) dropping computations, (ii) dropping synchronizations, and (iii) reducing the number of bits used to implement individual data elements. The goal behind data access dropping/skipping is to drop the right number of data accesses to maximize the performance/energy benefits and at the same time remain within the limits of "acceptable output inaccuracy" (output quality). Clearly, the latter is a function of the application/workload characteristics and can sometimes even change from one execution environment (e.g., user constraints, input) to another, for the same application program. This paper is based on a simple yet important observation:

> *If we are allowed to drop a certain amount of data accesses so that the program's output quality is still acceptable, to maximize performance/energy benefits, we may want to drop the "costliest" data accesses first.*

Motivated by this, first, we perform a quantitative analysis of the tradeoff between performance/energy benefits and inaccuracy (QoS) under varying amounts of data access skipping. The goal of this analysis is to demonstrate the importance of dropping the costliest data accesses (as opposed to, say, "randomly" selecting the data accesses to drop). We also investigate the variation in the program output qualities of two different executions of the same program, when in both the executions the same number of (but different) data accesses are dropped. Our results show that this variation can be quite high. Motivated by this, we then propose a "program slicing" based strategy that drops the right set of data accesses to guarantee the inaccuracy bound specified by the user/programmer. Unless stated otherwise, *when we say "a data access is dropped", we mean that the corresponding access is not performed and instead a value of "0" is assumed for that access.* Later, we discuss an alternate strategy (based on application profiling) to supply the missing values.

3 CONCLUDING REMARKS

This paper makes two main contributions. First, it investigates the potential benefits of a form of approximate computing that drops/skips select data accesses in the executions of multithreaded workloads on merging manycore systems. The unique aspect of this evaluation is its "architecture awareness". That is, given a bound on program output error (inaccuracy), we quantify the benefits of dropping the costliest data accesses (in the target architecture), as opposed to dropping data accesses randomly. Our experiments with ten different multithreaded workloads indicate that being architecture aware in dropping data accesses pays off, resulting in 27% additional performance improvement (on average) over randomly dropping the same number of data accesses. Second, it presents a program slicing-based approach that identifies the set of data accesses to drop such that (i) the resulting performance/energy benefits are maximized and (ii) the execution remains within the error (inaccuracy) bound specified by the user. Our experiments with this approach reveal 8.8% performance improvement and 13.7% energy saving (on average) are possible when we set the error bound to 2%, and these improvements increase to 15% and 25%, respectively, when the error bound is raised to 4%. Our ongoing work includes extending the proposed approach to nonvolatile memory (NVM) based systems.

ACKNOWLEDGMENT

We thank Murali Annavaram for shepherding our paper. We thank the anonymous reviewers for their feedback. This research is supported in part by NSF grants #1526750, #1763681, #1439057, #1439021, #1629129, #1409095, #1626251, #1629915, and a grant from Intel.

REFERENCES

[1] Han, J., and Orshansky, M. Approximate computing: An emerging paradigm for energy-efficient design. In *18th IEEE European Test Symposium (ETS)* (2013).
[2] Kandemir, M., Zhao, H., Tang, X., and Karakoy, M. Memory Row Reuse Distance and Its Role in Optimizing Application Performance. In *SIGMETRICS* (2015).
[3] Kayiran, O., Jog, A., Pattnaik, A., Ausavarungnirun, R., Tang, X., Kandemir, M. T., Loh, G. H., Mutlu, O., and Das, C. R. uC-States: Fine-grained GPU Datapath Power Management. In *PACT* (2016).
[4] Kislal, O., Kotra, J., Tang, X., Kandemir, M. T., and Jung, M. Enhancing computation-to-core assignment with physical location information. In *PLDI* (2018).
[5] Samadi, M., Jamshidi, D. A., Lee, J., and Mahlke, S. Paraprox: Pattern-based approximation for data parallel applications. In *ASPLOS* (2014).
[6] Samadi, M., Lee, J., Jamshidi, D. A., Hormati, A., and Mahlke, S. Sage: Self-tuning approximation for graphics engines. In *MICRO* (2013).
[7] Stanley-Marbell, P., and Rinard, M. Efficiency limits for value-deviation-bounded approximate communication. *IEEE Embedded Systems Letters* (2015).
[8] Tang, X., Kandemir, M., Yedlapalli, P., and Kotra, J. Improving Bank-Level Parallelism for Irregular Applications. In *MICRO* (2016).
[9] Tang, X., Kislal, O., Kandemir, M., and Karakoy, M. Data movement aware computation partitioning. In *MICRO* (2017).
[10] Venkataramani, S., Chippa, V. K., Chakradhar, S. T., Roy, K., and Raghunathan, A. Quality programmable vector processors for approximate computing. In *MICRO* (2013).