Self-Adaptive Memory Approximation: A Formal Control Theory Approach

Biswadip Maity, Majid Shoushtari, Amir M. Rahmani, *Senior Member, IEEE*, and Nikil Dutt, *Fellow, IEEE*

***Abstract*—Memory approximation enables trading off quality/accuracy for performance or energy gains. Traditionally, application programmers are burdened with the difficult task of setting memory approximation knobs to achieve the desired quality of service (QoS). Our self-adaptive approach for memory approximation eases the programmer's burden: simply specify the desired quality as a goal, with the system deploying a formal control-theoretic approach to tune the memory approximation knobs and deliver a guaranteed QoS. We model quality configuration tracking as a formal quality control problem, and outline a system identification technique that captures memory approximation effects with variations in application input and system architecture. Preliminary results show that we can alleviate the programmer's burden of manual knob tuning for on-chip memory approximation. When compared with a manual calibration scheme we achieve 3× improvement in average settling time and up to 5× improvement in best case settling time.**

***Index Terms*—Approximate computing, control theory, memory hierarchy, system identification.**

I. Introduction

THE GROWING literature in approximate computing for applications resilient to some level of imprecision shows that we can achieve higher performance and energy efficiency by trading off an acceptable loss in precision. Since memories continue to dominate energy consumption and pose a bottleneck for performance, memory approximation has the potential to yield significant gains in performance and energy. Prior work on memory approximation required programmers to *set approximation knobs* statically to appropriate values by trial and error in order to reach the desired quality of service (QoS) [1]–[3]. There is no mechanism to specify the required quality and have the system to generate the knobs to achieve the target automatically.

In this letter, we make the following contributions.

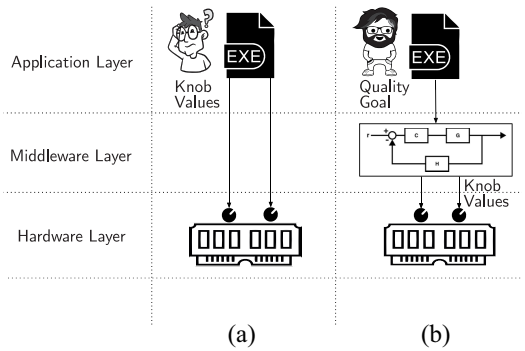

Fig. 1. Open-loop knob settings versus closed-loop quality control.

1) We propose a control-theory-based approach where developers only specify a target QoS metric and the system uses a formal control-theoretic approach to tune the memory reliability knobs of a quality-configurable memory to guarantee the desired QoS.
2) We outline how to develop a system model using the system identification theory for the memory components and how they react to different approximation settings using a statistical black-box modeling technique.
3) Using the developed system model, we design a controller that observes the application's behavior at fixed epochs and tunes knobs automatically to deliver the desired QoS despite changing workloads and system variations.

We show that our proposed methodology is able to maintain the QoS when operating on parts of the memory subsystem and can reach the required behavior much faster than a manual calibration scheme without any tuning by the programmer. To the best of our knowledge, this is the first work to introduce the idea of using a formal control-theory-based approach for memory approximation.

II. Motivation

The QoS delivered via approximations is affected by multiple parameters, including the configuration of the memory hierarchy, the application input and temporal relationship between inputs, and making manual tuning extremely challenging.

As an example, we illustrate the challenge raised by varying application load that affects the delivered QoS. Traditionally, developers profile an application and determine the best possible knob for a given target and expect the same QoS for a

Manuscript received July 2, 2019; revised August 22, 2019; accepted August 29, 2019. Date of publication September 11, 2019; date of current version May 27, 2020. This work was supported in part by NSF under Grant CCF-1704859. This manuscript was recommended for publication by A. Easwaran. *(Corresponding author: Biswadip Maity.)*

B. Maity, M. Shoushtari, and N. Dutt are with the Department of Computer Science, University of California at Irvine, Irvine, CA 92697 USA (e-mail: maityb@uci.edu; anamakis@uci.edu; dutt@uci.edu).

A. M. Rahmani is with the Department of Computer Science, University of California at Irvine, Irvine, CA 92697 USA, and also with the Institute of Computer Technology, TU Wien, 1040 Vienna, Austria (e-mail: amirr1@uci.edu).

Digital Object Identifier 10.1109/LES.2019.2941018

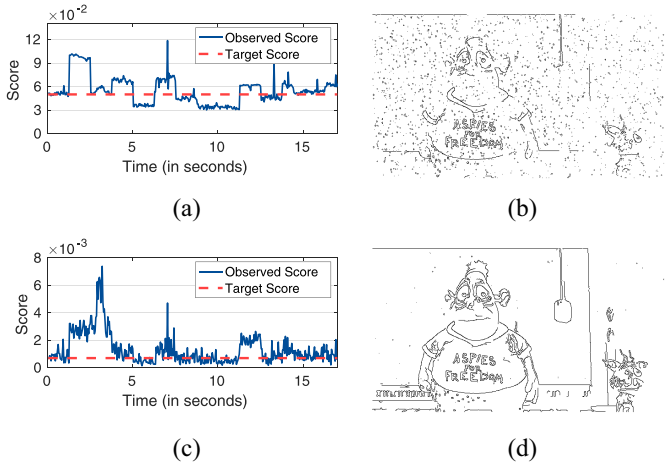

Fig. 2. (a) and (c) Variation of the quality of the edge detection in a video scene when the BER is constant. (b) and (d) Frame in the video with the expected score for the given setting of BER. (a) BER = 1E-3, expected score = 0.05. (b) Score = 0.05. (c) BER = 1E-5, expected score = 0.001. (d) Score = 0.001.

given setting of the knob throughout the application's lifetime. However, fixed knob settings result in varying QoS in the face of changing workloads, as shown in Fig. 2(d). It shows the quality of edge detection in a video composed of multiple scenes when the write bit error rate (BER) is kept constant. Fig. 2(a), (c), and (d) shows significant variations in quality across different inputs, demonstrating the drawback of an open-loop set-once-and-execute approach. This traditional open-loop approach suffers several drawbacks.

1) It is difficult to model an under-designed memory in order to measure the output accuracy at different settings. Temporal faults that are variability-induced, temperature-induced, etc., cannot be modeled easily. Unlike software approximation strategies that are easier to evaluate, hardware approximation requires rigorous runtime tuning. Application profiling to generate fixed parameter knobs cannot yield a consistent quality metric.
2) They make approximation decisions based on average or worst-case input behavior. These techniques rely on training with inputs that attempt to represent real-world inputs, which are difficult to achieve in practice. Laurenzano *et al.* [4] have shown that accuracy of approximate programs depends heavily on program input.
3) Different components in the memory subsystem react differently to each workload due to differences in memory access patterns. Although techniques such as memory profiling can help to estimate the knobs for a given system, once the application is ported to a different system, the programmer needs to manually recalibrate the knobs in order to achieve the desired quality.

With our approach, the programmer—instead of setting the knobs directly—only needs to specify the desired quality and does not need to profile the application for each target system. This not only eases the task of the programmer but also makes the approximation portable across different computing platforms and memory technologies as shown in Fig. 1.

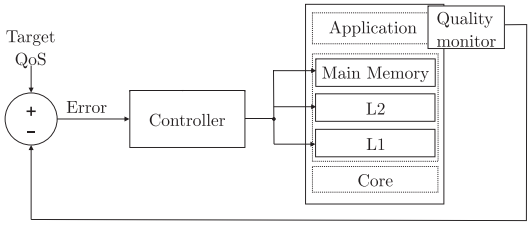

Fig. 3. Closed-loop approach for tuning memory approximation knob(s).

III. Proposed Approach

In this section, we show that despite the randomness of errors introduced into the execution of programs due to memory approximation, a formal control-theoretic technique can capture the system dynamics effectively. The effectiveness is demonstrated by quality control of the program even in the presence of stochastic (nondeterministic) behaviors.

In our case, the target *system* is composed of both hardware (with quality-configurable memory) and software (the application running on the hardware) as shown in Fig. 3. Depending on the memory technology, there can be one or more knobs that can tune the degree of approximation in the memory subsystem. Some examples of control knobs are: 1) voltage in SRAM memory; 2) refresh rate in DRAM; and 3) read/write current amplitude in nonvolatile memories such as STT-MRAM. To make our experiments technology-independent, we use BER as our knob which is the probability with which each bit flips during a memory read/write operation. With a predetermined frequency, the quality monitor routine measures the current quality of the output and compares it against the quality goal. A positive difference means there is still room to relax the reliability requirements of the memory and the controller accordingly sets a more aggressive knob setting. A negative difference indicates that the quality has degraded more than what was intended. The controller accordingly sets a more conservative knob setting.

A. System Identification

The use of statistical or black-box methods to construct models of a system is known as the *system identification*. A common practice to design a feedback system using a controller is to extract the dynamic model of complex systems through system identification theory [5]–[7]. By varying BER experimentally, data is collected to see the effects of BER on the measured output (score) for each memory component. A waveform with a step-pattern is applied at the inputs, and the output is continuously monitored. The monitoring process is repeated for several video streams and average output quality at each BER knob used to model the relationship.

The relationship between control inputs and measured outputs can be specified using the linear differential equations. A simple first-order linear difference equation can be approximated as: $y(k+1) = ay(k) + bu(k)$, where a and b are the parameters that can be identified using parameter estimation methods, such as *least square regression*. This equation represents a first-order model where the next output depends only on the inputs and outputs from one time-unit in the past.

The control inputs and measured outputs are used to estimate a linear-parametric model through a transfer function. If we are given a system with transfer function $G(z)$ and input $U(z)$, then the output of the system can be described as $Y(z) = G(z) \times U(z)$. Transfer functions have properties, such as stability, steady-state gain, settling time, and maximum overshoot. We use the system identification toolbox in MATLAB to estimate the transfer functions when BER control knob is applied on L1 data cache, L2 cache, and DRAM. The estimated models can be found in [8]. For all the components, we used a second-order model with two poles ($n_p = 1$) and one zero ($n_z = 1$).

B. Application and Quality Metric

In this letter, we target streaming applications which have temporal dependencies between consecutive inputs. The application is given a sequence of inputs, and the results from processing previous inputs can be used to adjust the knobs for successive inputs. The adjustment is possible due to the correlation of the inputs as well as the temporal behavior of the memory errors. We use the canny video edge detection [9] as our case study. Edge detection is the process of identifying sharp changes in image brightness. For video processing, edge detection is often conducted on a frame-by-frame basis independently. Adjacent frames of a scene in a video have temporal similarity, and it allows the controller to adjust the quality based on the history of the system.

The quality measurement method is application dependent, and normally the programmer provides a software routine for measuring it at runtime. In many cases, this quality measurement would require computing the precise and approximate versions of the output for comparison [10], [11]. We use miss-classification error (ME) as our QoS metric, that is, the ratio of the total number of pixels mistakenly classified as edge/nonedge to the total number of pixels in the frame. To evaluate the performance of the method, the settling time is computed. Settling time is the time for the output to reach the target value after a change in one of the inputs. We consider the output to have settled when it is within 2% of its target value.

C. Controller Design

Our system is a simple single-input–single-output (SISO) control system with BER as a control input and edge detection miss-classification rate as measured output. We use a proportional–integral (PI) controller to control this system. The proportional term refers to the fact that the controller output is proportional to the amplitude of error signal, while *integral* indicates that the controller output is proportional to the integral of all past errors [7]. The PI control law has the form

$$u(k) = u(k-1) + (K_P + K_I)e(k) - K_P e(k-1) \qquad (1)$$

where K_P and K_I denote the coefficients for the proportional and integral terms, respectively. *Controller design* is a mature field which utilizes many tools that provide off-the-shelf controllers. We use MATLAB PID tuner toolbox to design our controllers. It is important to note that although derivative control law is helpful to add predictability to the controller, stochastic variations in the system output may cause inaccuracy in the controller. This issue becomes more severe in computer systems as they commonly have a significant stochastic component. Therefore, for computer systems, PI controllers are preferred over PI-derivative (PID) controller [7]. PI control benefits from both integral control (zero steady-state error) and proportional control (fast transient response).

IV. Evaluation and Results

We evaluate the performance of the self-adaptive system by comparing it with a manual recalibration scheme. In the self-adaptive system, we define a target QoS accuracy while the middleware controls the knobs automatically. We expect: 1) the system to adapt to changes in application input automatically and 2) the settling time to be significantly less than the manual recalibration scheme. More details about the experimental setup and system dynamics can be found in [8].

A. System Overview

To simulate the behavior of a system with approximate memory, we developed a Sniper-based [12] memory fault injector (FI).[1] This FI can inject faults into read/write operations of the memory hierarchy (e.g., cache, TLB, and DDR). To inject faults only into the noncritical data objects of the program, the source code of the program is annotated with `add_approx()` and `remove_approx()` methods to declare the address of those data objects in the program. These methods are called in the program at appropriate places and are captured by the FI. The FI records these addresses into a table. During the execution, it instruments all the memory accesses. If the virtual address of the access falls into the any of the given address boundaries, it attempts to inject a fault into the part of data referenced by that memory access. The controller, which is implemented in the middleware, is capable of receiving the results from quality monitors at runtime and set the read/write BER knobs in the simulation framework.

B. Control Experiment—Manual Recalibration

The manual scheme measures the difference between the desired quality and current quality. If this difference is within $\pm 10\%$ it does not change the knobs. Otherwise, it changes the knob in one direction with fine-grained steps until the quality returns to the acceptable quality region. To recalibrate dynamically, it multiplies the steps with the *logarithm* of difference in quality.

C. Comparison Between Self-Adaptive System and Manual Calibration

Quality tracking is simulated using different control mechanisms in Figs. 4 and 5 and the system's performance is evaluated. The figures show how the feedback loop operates in practice for video inputs. The red dashed curve shows the quality goal (or expected behavior). The blue curve shows achieved quality (or observed behavior) for PI control. The orange curve shows the achieved quality for manual control.

The blue curve in Fig. 4 shows the performance of a self-aware system equipped with a PI controller in tracking target quality for L1 data (L1D) cache write errors. The tracking

[1]Code repository at https://github.com/duttresearchgroup/memapprox-control.

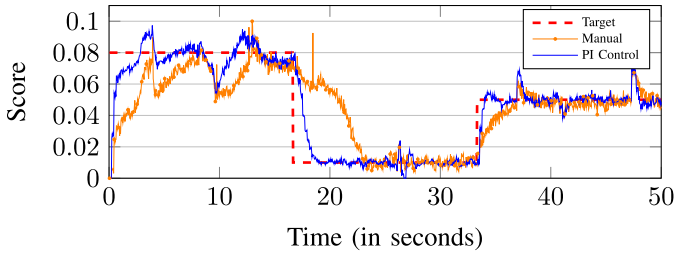

Fig. 4. Runtime quality tracking for L1 data cache write errors. Self-evaluation done every five frames.

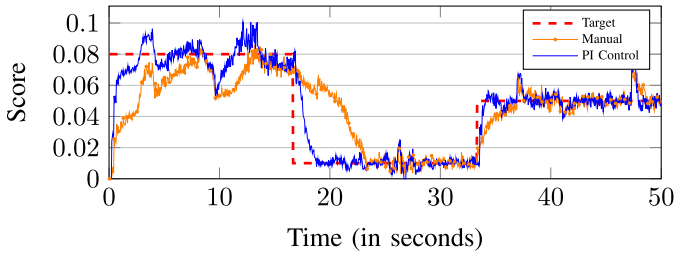

Fig. 5. Runtime quality tracking for L2 cache write errors. Average settling time for PI control = 2.04 s, and average settling time for manual control = 6.25 s.

can be performed with an average settling time of 1.96 s. The orange curve shows the performance of a system with a manual recalibration scheme. In the manual scheme, the average settling time is 6.13 s, making the settling time 3× faster for the L1D writes when using the self-adaptive system. Fig. 5 shows a similar evaluation where we approximate the memory writes of L2 cache instead of L1D. For L2 write errors, the average speedup of settling time is also about 3×. In the best case, both L1D and L2 write errors have a speedup in settling time of 5×.

Our controller is capable of effectively tracking the quality when we approximate the on-chip memory. However, there remain challenges when performing this approximation on DRAM. Our initial investigation suggests that DRAM is more tolerant to errors because for the same score, the expected BER knob value of DRAM is almost 5× more than on-chip BER knob values. We note that our attempt to track the quality using our self-adaptive approach was unsuccessful; we hypothesize that our controller cannot take advantage of the temporal similarity across frames since DRAM access patterns are nonuniform, therefore, the quality monitor, being the only feedback is insufficient in controlling the target effectively. This highlights the need to investigate further opportunities for using proposed method in the context of off-chip memory accesses.

Energy Measurement: In the simulation of quality-configurable memory, the underlying technology is abstracted by choosing BER as the control knob. Since there are models for each technology which maps the technology-dependent control knob to BER, the models can be incorporated in the infrastructure by adding another lookup-table or an equation which translates between BER and the technology-dependent control knob. A summary of the mappings and exact models can be found in [8]. An example equation of write energy (pJ) for STT-MRAM writes, as obtained from the model [13], is

$$\min\{-4.836 \times \log_{10}(\text{BER}) - 1.022, 36\}. \quad (2)$$

The energy consumed is measured by invoking McPAT within the simulation infrastructure. Using (2) for L1 data cache, an energy savings of 10.5% is obtained for PI control, and 9.4% for manual control.

PI Controller Overhead: We measure the PI controller overhead at runtime as 20 000 instructions. The average number of instructions for processing a frame over 100 frames is 42 836 000. The PI controller overhead is, therefore, 0.04% on average.

V. Conclusion

We presented an approach based on the formal control theory to design a self-aware system which can control the quality of applications running on systems with approximate memories. We showed that our system adapts to changing workloads as well as takes into account the variations in underlying architecture. Since, the current implementation of middleware uses an SISO controller, we approximate the components one at a time. While, in this case study, we utilize an SISO controller, a multiple-input–multiple output (MIMO) controller could also be used [14]. MIMO controllers may be more effective—due to either multiple knobs per memory component (i.e., STT-MRAM read and write current) or the controller tuning various individual memory components in the memory hierarchy, however, the main challenge of using MIMO control in approximate computing would be the system identification phase.

References

[1] M. Shoushtari, A. BanaiyanMofrad, and N. Dutt, "Exploiting partially-forgetful memories for approximate computing," *IEEE Embedded Syst. Lett.*, vol. 7, no. 1, pp. 19–22, Mar. 2015.

[2] S. Liu, K. Pattabiraman, T. Moscibroda, and B. Zorn, "Flikker: Saving DRAM refresh-power through critical data partitioning," in *Proc. ASPLOS*, 2011, pp. 213–224.

[3] A.-M. Monazzah, M. Shoushtari, A. Rahmani, and N. Dutt, "QuARK: Quality-configurable approximate STT-MRAM cache by fine-grained tuning of reliability-energy knobs," in *Proc. ISLPED*, Taipei, Taiwan, 2017, pp. 1–6.

[4] M. A. Laurenzano, P. Hill, M. Samadi, S. Mahlke, J. Mars, and L. Tang, "Input responsiveness: Using canary inputs to dynamically steer approximation," in *Proc. PLDI*, 2016, pp. 161–176.

[5] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[6] L. Lennart, "Black-box models from input–output measurements," in *Proc. IMTC*, 2001, pp. 138–146.

[7] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. Hoboken, NJ, USA: Wiley, 2004.

[8] B. Maity, M. Shoushtari, A. Rahmani, and N. Dutt, "Simulation infrastructure and system dynamics of quality configurable memory," Center Embedded Cyber Phys. Syst., Univ. California at Irvine, Irvine, CA, USA, Rep. TR 19-03, Jul. 2019.

[9] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Jun. 1986.

[10] W. Baek and T. M. Chilimbi, "Green: A framework for supporting energy-conscious programming using controlled approximation," in *Proc. PLDI*, 2010, pp. 198–209.

[11] Aurangzeb and R. Eigenmann, "Harnessing parallelism in multicore systems to expedite and improve function approximation," in *Proc. LCPC*, 2016, pp. 88–92.

[12] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 3, 2014, Art. no. 28.

[13] A. Ranjan, S. Venkataramani, X. Fong, K. Roy, and A. Raghunathan, "Approximate storage for energy efficient spintronic memories," in *Proc. DAC*, 2015, Art. no. 195.

[14] T. Mück, B. Donyanavard, K. Moazzemi, A. M. Rahmani, A. Jantsch, and N. D. Dutt, "Design methodology for responsive and robust MIMO control of heterogeneous multicores," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 4, pp. 944–951, Oct.–Dec. 2018.