Quality-Configurable Approximate Memory Hierarchy: A Formal Control Theory Approach

Majid Shoushtari
Department of Computer Science
University of California, Irvine, CA
anamakis@uci.edu

Amir M. Rahmani
TU Wien, Vienna, Austria
University of California, Irvine, CA
amirr1@uci.edu

Nikil Dutt
Department of Computer Science
University of California, Irvine, CA
dutt@uci.edu

Abstract

This paper proposes utilizing formal control theory to control the quality of streaming approximate applications running on a system with quality-configurable memory by tuning memory reliability knob(s).

1 Introduction

Approximate computing trades accuracy for performance or efficiency, exploiting the fact that many applications are robust to some level of imprecision. One of the primary candidates in the system for approximation is the memory subsystem.

Previous work has focused mainly on open-loop and profile-guided approaches for *setting approximation knobs* statically to appropriate values [2, 7, 13] (Fig. 1(a)). Instead of putting the burden of setting approximation knob(s) on the system designer or the programmer, we aim at automatically *controlling the quality* by utilizing a feedback controller in the middleware layer that finds the optimal knob values at runtime (Fig. 1(b)) depending on the requested quality for the application.

Profile-guided open-loop approaches suffer also from two major disadvantages: (1) They make approximation decisions based on average or worst-case input behavior. Laurenzano et al. [4] have shown that the accuracy of approximate programs depends heavily on program input. (2) 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.

In this work, we target streaming applications in which the application is given a sequence of inputs and the results from processing previous inputs can be used to adjust the knobs for the successive inputs because of the correlation of the inputs as well as the temporal behavior of the memory errors. Formal control theory lends itself well to this kind of applications where the history and trend of the previous inputs can be utilized to make predictions for future inputs. In other words, when the system dynamics can be formulated using difference/differential equations.

Several work has done dynamic control for approximation [1, 11, 14]; this work aims to fill that role with control theory. We believe that despite the randomness of errors introduced into the execution of the program because of the memory approximation, the system dynamics can be captured in way that a formal control-theoretic technique can effectively control the quality of the program even in presence of stochastic (non-deterministic) behaviors. The systematic design of closed-loop systems needs an ability to quantify the effect of control inputs (e.g., approximation knobs) on measured outputs (e.g., quality of the program's output), both of which may vary with time. Identifying this model (i.e., dynamic behavior) is central to leveraging control theory for managing a system. Our solution to this problem employs a controller to construct a closed-loop feedback control mechanism. This controller monitors the error (i.e., difference between target output quality and measured output quality) and applies a correction accordingly.

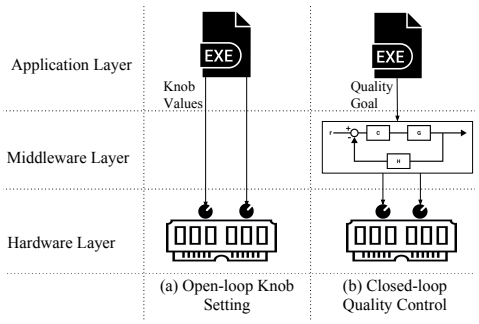


Figure 1. Open-loop knob settings vs. closed-loop quality control.

2 Problem Formulation

2.1 Application Class

We target streaming applications where a stable computation pattern is applied on consecutive inputs. Usually these applications are stateless. In the context of approximate computing, this means that the quality degradation of kth input is purely dependent on the hardware errors happened during processing that input.

2.2 Monitoring Quality at Runtime

In order to construct a closed loop feedback mechanism and adjust the memory approximation knob(s) at runtime, the system needs to occasionally sample the current quality of the output with the current knob(s) settings. This quality measurement method is application dependent and normally the programmer provides a software routine to measure it at runtime. In many cases, this quality measurement would require computing the precise and approximate versions of the output for comparison.

2.3 Memory Approximation Knob(s)

Depending on the approximation strategy used and the memory technology, there is always one or more knobs that can tune the degree of approximation storage. For example, in SRAM memories, voltage is the main knob. If the approximation strategy is to skip some of the memory accesses – often stores – or predicting the value of load operations, the percentage of such operations could be the knob that tuning of which would adjust the final quality of the output. The ultimate goal is to set the most aggressive knob setting that satisfies the quality goal in order to gain the highest energy savings possible.

3 Quality Control with Feedback Control Theory

In control theory, a controlled system is represented as a feedback control loop as in Fig. 2(a). At the epoch k, the controller reads the *measured output y(k)* of a system in *state x(k)*, compares it against the target value y_{ref}, and based on the difference (or error *e(k)*), generates the *control input u(k)* to actuate on the system and reduce the error.

In our case, the system is composed of both quality-configurable memory as well as the software running on a system with this memory. The control input to the system is the value of a knob and the measured output of the system is the quality of the generated output.

The closed loop approach for tuning knobs is shown in Fig. 2(b). The application runs on a processor with a quality-configurable memory. With a pre-determined frequency, the quality monitor routine measures the current quality of the output. This quality is compared against the quality goal. A positive difference means there is still room to relax the

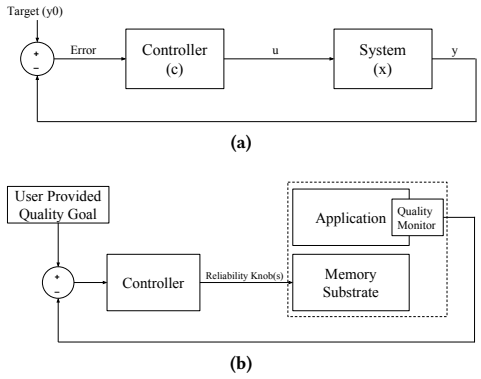


Figure 2. Closed loop approach for tuning memory approximation knob(s).

reliability requirements of the memory and the controller accordingly sets a more aggressive knob setting. A negative difference means that the quality has degraded more than what was intended. The controller accordingly sets a more conservative knob setting.

To formally control such a system, a model of the system dynamics that describes the relation between control input(s) (i.e., knob(s)) and the measured output (i.e., the quality of the program's output) as a function of time epoch, is needed.

A common practice to extract the dynamic model of complex systems is through System Identification Theory [5, 6] where we experimentally collect input-output data and use black-box identification methods to isolate the deterministic and stochastic components of the system. The relationship between control inputs and measured outputs can be specified using linear difference equations. More precisely, it relates current and past outputs to current and past inputs. A simple first order linear difference equation can be approximates as: $y(k+1) = ay(k) + bu(k)$, where a and b are parameters that can be identified using parameter estimation methods such as *least square regression*. This is a different use case of machine learning as the system is first exercised with $u(k)$ to observe $y(k)$, and then $y(k+1)$ which is the shifted version of $y(k)$ is used to fit a regression in a three-dimensional space. In this way, we can build a model entirely from true data for arbitrarily complex systems. In our design, we follow this approach and show that *quality configuration (i.e., tracking) in memory-oriented approximate computing can be modeled as a formal quality control problem*, which can be then addressed by using classical off-the-shelf controllers.

4 Case Study: Video Edge Detection

We use canny video edge detection as our case study. The temporal similarity between adjacent frames of a scene in a video allows the controller to adjust the quality based on the history of the system from previous frames. We use miss-classification error as our quality metric, that is the ratio of total number of pixels mistakenly classified as edge/non-edge to the total number of pixels in the frame.

For simplicity, we assume that all the memory accesses hit the memory substrate in which we are interested to actuate the error rate. We choose write reliability as the tuning knob as the write current amplitude which is shown in previous work to be an appropriate approximation knob for spin-transfer torque memories [7, 10, 12], and there is a relationship between the amount of the current applied for write operations and the write error probability.

4.1 System Identification

In order to design a controller for the system, we need to model the system first. The first step towards that is to generate test waveforms from training applications for system identification. A test waveform contains a series of samples for controller inputs and outputs for a training application, and should exercise as many input permutations as possible. The system dynamics is exercised often by applying a staircase waveform to the control input (e.g., write bit error rate). Such staircase would stimulate system behaviour in response to various levels of control input. In our work, we change write bit error rate from 10E-7 to 10E-3 with steps of 10E-7.

In this method, training sets use varying frequency (e.g., a set of out-of-phase staircase signals for the control input) in order to isolate the deterministic and stochastic aspects of the system. This model is then evaluated to predict the expected data from the identified system. Abnormal behaviour from this model can raise a flag that the controller to be designed from this model might be inaccurate. We use MATLAB's system identification toolbox for this process. Figure 3 shows the result of system identification for canny when a series of 2400 frames are inputted. It can be seen that the predicted model closely fits the measured data.

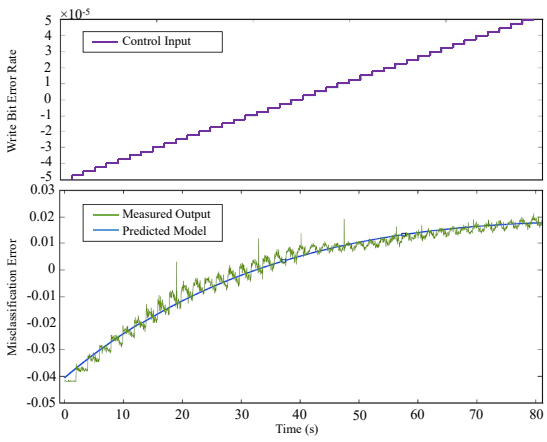


Figure 3. Predicted Model vs. Measured Output.

4.2 Controller

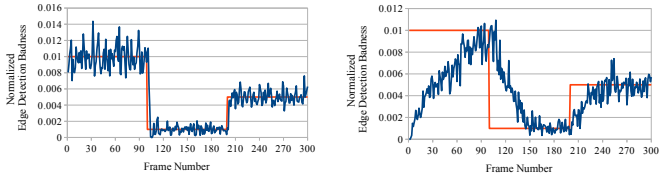
Our system is a simple single-input-single-output (SISO) control system with write bit error rate as control input and edge detection miss-classification rate as measured output. [1] We use a proportional-integral (PI) controller to control this system. The proportional term refers to the fact that the 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 [3]. The PI control law has the form:

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

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 and deploy our controllers. PI control benefits from both integral control (zero steady-state error) and proportional control (fast transient response). In most computer systems a first-order PI controller provides rapid response and is sufficiently accurate [3].

4.3 Comparison

Figure 4 compares the performance of a PI controller in tracking target quality with a manual calibration scheme. The manual scheme measures the difference between the desired quality and the current quality. If the difference is within -+10% it does not change the knobs. Otherwise it changes the knob in one direction with certain fine-grained steps until the quality returns back to the acceptable quality region.



(a) pi controller **(b)** manual calibration

Figure 4. Comparing PI controller with manual step-wise re-calibration similar to [1].

5 Concluding Remarks and Future Work

We presented an approach based on the formal control theory in order to control the quality of applications running on systems with approximate memories. Future work will look into using this mechanism for a set of approximate streaming applications. Also, a detailed analysis for the energy trade-off achievable with this approach is required. We also intend to find use cases where a MIMO controller could be more effective because of the number of quality metrics that can be controlled and/or the number of memory components that can be independently tuned. Our recent work [8] presents a methodology for designing robust and responsive MIMO controllers for manycore, while in [9], we present a hierarchical supervisory control approach which leverages MIMO controllers for resource management in manycore systems. We plan to investigate their effectiveness in the context of quality control in approximate computing.

[1] Note that while in this case study we utilize a SISO controller, we also acknowledge the possibility of using a multiple-input-multiple output (MIMO) controller for other systems that may provide more than one output quality metric and memories that provide more than one to tune – either because there are multiple knobs per memory component (i.e., STT-MRAM read and write current) or the controller is tuning multiple individual memory components in the memory hierarchy.

References

[1] Woongki Baek and Trishul M. Chilimbi. 2010. Green: A Framework for Supporting Energy-conscious Programming Using Controlled Approximation. In *Proc. of PLDI*.

[2] S. Liu et al. 2011. Flikker: Saving DRAM Refresh-power Through Critical Data Partitioning. In *Proc. of ASPLOS*.

[3] Joseph L. Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M. Tilbury. 2004. *Feedback Control of Computing Systems*. John Wiley & Sons.

[4] Michael A. Laurenzano, Parker Hill, Mehrzad Samadi, Scott Mahlke, Jason Mars, and Lingjia Tang. 2016. Input Responsiveness: Using Canary Inputs to Dynamically Steer Approximation. In *Proc. of PLDI*.

[5] Lennart Ljung. 1999. *System Identification: Theory for the User*. Prentice-Hall, Inc.

[6] Lennart Ljung. 2001. Black-box Models from Input-output Measurements. In *Proc. of IMTC*.

[7] Amir-Mahdi Monazzah, Majid Shoushtari, Amir Rahmani, and Nikil Dutt. 2017. QuARK: Quality-configurable Approximate STT-MRAM Cache by Fine-grained Tuning of Reliability-Energy Knobs. In *Proc. of ISLPED*.

[8] Tiago Muck, Bryan Donyanavard, Kasra Moazzemi, Amir M. Rahmani, Axel Jantsch, and Nikil Dutt. 2018. Design Methodology for Responsive and Robust MIMO Control of Heterogeneous Multicores. *IEEE Transactions on Multi-Scale Computing Systems* (2018).

[9] Amir M. Rahmani, Bryan Donyanavard, Tiago Muck, Kasra Moazemmi, Axel Jantsch, Onur Mutlu, and Nikil Dutt. 2018. SPECTR: Formal Supervisory Control and Coordination for Many-core Systems Resource Management. In *Proc. of ASPLOS*.

[10] Ashish Ranjan, Swagath Venkataramani, Xuanyao Fong, Kaushik Roy, and Anand Raghunathan. 2015. Approximate Storage for Energy Efficient Spintronic Memories. In *Proc. of DAC*.

[11] Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. 2013. SAGE: Self-tuning Approximation for Graphics Engines. In *Proc. of MICRO*.

[12] Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. 2013. Approximate Storage in Solid-state Memories. In *Proc. of MICRO*.

[13] M. Shoushtari, A. BanaiyanMofrad, and N. Dutt. 2015. Exploiting Partially-Forgetful Memories for Approximate Computing. *IEEE Embedded Systems Letters* (2015).

[14] Xin Sui, Andrew Lenharth, Donald S. Fussell, and Keshav Pingali. 2016. Proactive Control of Approximate Programs. In *Proc. of ASPLOS*.