SANDIA REPORT

SAND2017-10350
Unlimited Release
Printed September 2017

Supersedes SAND1901-0001
Dated January 1901

A Case Study on Neural Inspired Dynamic Memory Management Strategies for High Performance Computing

Craig M. Vineyard, Stephen J. Verzi

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

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

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology and Engineering Solutions of Sandia, LLC.

SAND2017-10350
Unlimited Release
Printed September 2017

Supersedes SAND1901-0001
dated January 1901

A Case Study on Neural Inspired Dynamic Memory Management Strategies for High Performance Computing

Craig M. Vineyard
Data-Driven & Neural Computing
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-9999
cmviney@sandia.gov

Stephen J. Verzi
Sys Research, Analysis, & Apps
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-9999
sjverzi@sandia.gov

Abstract

As high performance computing architectures pursue more computational power there is a need for increased memory capacity and bandwidth as well. A multi-level memory (MLM) architecture addresses this need by combining multiple memory types with different characteristics as varying levels of the same architecture. How to efficiently utilize this memory infrastructure is an unknown challenge, and in this research we sought to investigate whether neural inspired approaches can meaningfully help with memory management. In particular we explored neurogenesis inspired resource allocation, and were able to show a neural inspired mixed controller policy can beneficially impact how MLM architectures utilize memory.

Acknowledgment

Thanks to Meghan Galiardi for helping to develop a mathemtical model of post cache memory access, to Jacob Hobbs for his contributions on adaptive strategies, to Benjamin Anthony for preliminary investigations into reinforcement learning, and Arun Rodrigues for his expertise in computer architecture.

Thank you John Wagner for serving as the Program Manager of this project.

Contents

List of Figures

Summary

As high performance computing architectures pursue more computational power there is a need for increased memory capacity and bandwidth as well. A multi-level memory (MLM) architecture addresses this need by combining multiple memory types with different characteristics as varying levels of the same architecture. How to efficiently utilize this memory infrastructure is an unknown challenge, and in this research we sought to investigate whether neural inspired approaches can meaningfully help with memory management.

In particular, we explored neurogenesis inspired resource allocation. Neurogenesis is a process by which new neurons are included in existing neural circuits. A mixed coding hypothesis posits that as an adaptive algorithm the neurogenesis process enables the encoding of novelty. The neurogenesis process may be applied to MLM management in a variety of ways. Highlighted in this report, analogous to a mixed coding, a mixed strategy of memory control policies can enable a more efficient memory mapping. This proof of concept case study on whether neural inspired approaches can meaningfully impact high performance computing provides a basis for a more fundamental understanding of the interplay between architectures and programs and additionally provides motivation for a more advanced reinforcement learning approach to learn sophisticated strategies. It is also likely that other high performance computing aspects can likewise employ neural inspired advantages.

Chapter 1

Introduction

As the field of computing has advanced over the last few decades, larger and larger computer systems have been developed to meet the demands of scientific computing and other fields. Various computer architectures have been explored to address these needs focusing upon parallelism and other computational advantages. As modern systems are striving towards exascale (a billion billion calculations per second), there is an increasing demand on memory bandwidth and capacity in conjunction with the compute power needed. Multi-level memory (MLM) architectures are being proposed as an architectural solution. As follows we introduce MLM and the implications this architecture introduces.

1.1 Multi-Level Memory (MLM)

Multi-Level Memory (MLM) is an architectural configuration which combines different memory technologies with different characteristics in a single system. For example, main memory can be comprised of two or more types of memory instead of conventional single-level DDR DRAM-only main memory. With different capabilities associated with the various memory types, higher-bandwidth memories are ideally able to serve the majority of accesses and low cost memory types provide the majority of the capacity. Figure 1.1 illustrates a two level hierarchy concept where the memory closer to the computational cores is intended to provide quicker access and the further memory on the right affords a larger capacity. The MLM paradigm is being pursued across a spectrum of levels of memory.

As proposed MLM architectures become more prominent, MLM partitioning within compute nodes will become a crucial aspect of achieving high-performance levels. However, simply establishing the increased capacity does not ensure increased performance. Next we briefly introduce a game theoretic paradox highlighting this concept and then describe the implications for MLM architectures.

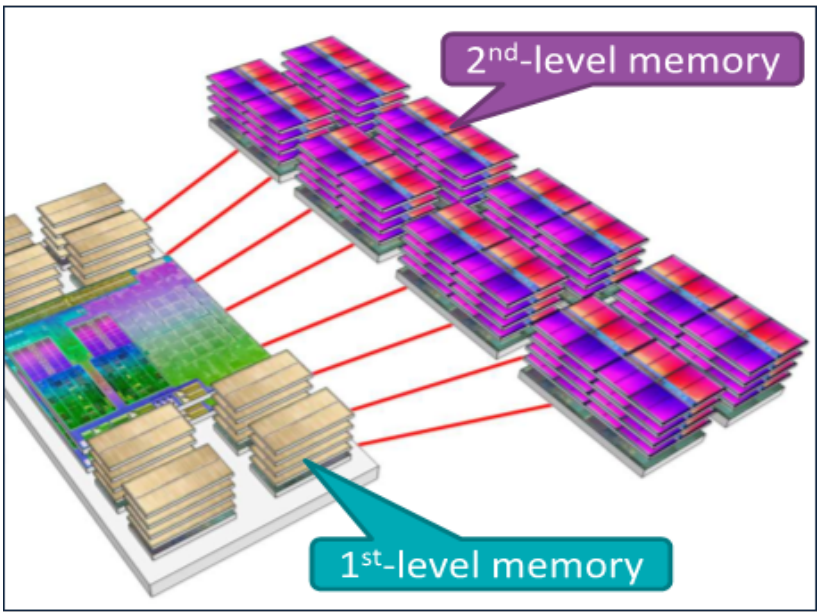


Figure 1.1. Illustration of a two level MLM hierarchy concept

1.2 Braess' Pardox

Mathematician Dietrich Braess presented a traffic flow paradox in 1968 [13] which is shown in Fig 1.2. In the simple scenario show on the left, there are four nodes with two paths (an upper and lower) to travel from start to end. Some path segments maintain a fixed cost regardless of how much traffic is present (represented by a cost of 1 as shown in the figure). Other path segments have costs dependent upon how much traffic the segment experiences. In this scenario, the optimal strategy is for half of the traffic to take each path so that everyone incurs an average travel delay of 1.5 (this is illustrated by the two dashed red arrows).

The network shown on the right half of the figure represents the same nodes, but includes the addition of a high bandwidth connection between intermediate nodes a and b. This path segment incurs zero cost regardless of how much traffic traverses the path (denoted by a cost of zero). The inclusion of this additional path is intended to improve the overall network infrastructure, but in reality unintentionally makes the average network flow worse overall. With the inclusion of this link, the optimal strategy for all network traffic to take the link from start to a, to then use the new link from a to b, and reach the end by the b to end path. This path shown in red on the right results in all traffic going through both bandwidth dependent links at the detriment of overall flow. This is Braess' paradox where what can be seen as an infrastructure improvement with good intentions can yield unintended consequences.

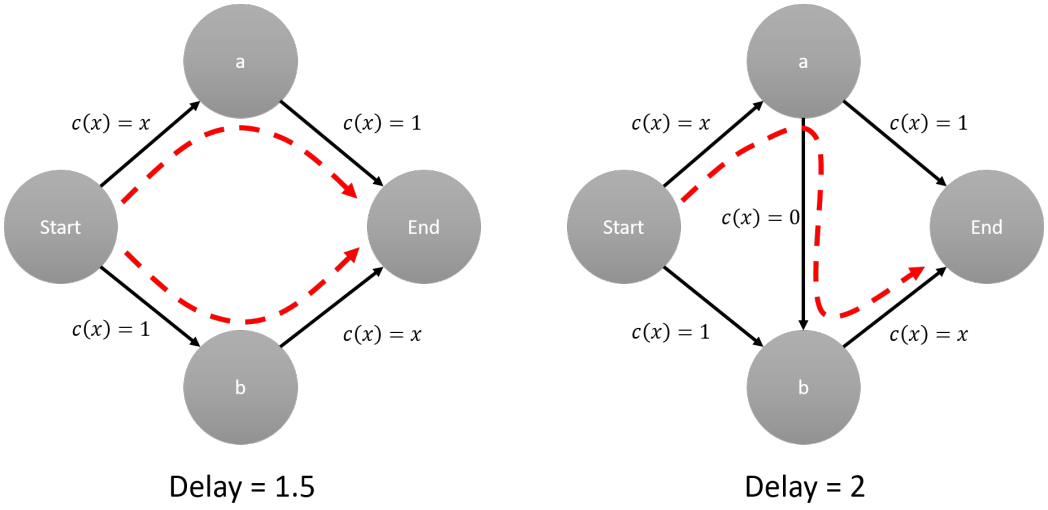


Figure 1.2. Illustration of Braess's Paradox in a Simple Network Flow

1.3 MLM Implications & Related Problems

While the inclusion of a variety of memory technologies in an architecture is a means of enabling immense bandwidth and capacity, as Braess' pardox identifies, the inclusion of increased capacity alone does not guarantee increased performance. In the case of MLM, the inclusion of a variety of memory technologies introduces the challenge of how to make use of the memory technologies as no established methods to dynamically manage this partitioning exist.

Intrinsically, the operations needed to perform a computation (instruction flow, data required, etc.) are not knowing *a priori* or there would be no need to perform the computation. Instead, this uncertainty introduces a resource allocation challenge as an intrinsic part of computation. In particular, with respect to MLM management, it is a resource allocation challenge of deciding what variables should reside in different memories over time.

Several fundamental computer science problems strive to address such challenges pertaining to resource allocation. For example, the classic Knapsack problem is a combinatorial optimization problem in which a subset of items with the maximum possible combined value can be collected within a fixed capacity knapsack [4]. This optimization problem is a NP-Hard computational complexity problem. But furthermore, in relating the problem to memory management, a MLM architecture further escalates the challenge to that of a time varying multi-knapsack problem in which the different knapsacks have different capacities (the different memory types) and the set of items to optimize over is temporally varying due to the computation program flow.

Another challenging but related problem is the Multi-Armed Bandit (MAB) from game theory [13]. The notional problem is that of selecting which slot machine to play from k possibilities so

as to maximize ones revenue. The outcome distributions of the various machines are unknown, and so it cannot be simply cast as an optimization problem. This resource allocation problem becomes selecting which option to select at each point in time. Analogously, selecting from possible memory controllers to execute or which program variables to store in a given memory location shows a relationship between MLM management and MAB problems. A common challenge these problems share is the explore-exploit tradeoff in which in the presence of uncertainty one must choose between continuing using the same selection (exploit) or to switch and potentially find a better outcome (explore).

With no obvious tractable solutions to these fundamental computer science resource allocation problems, instead next we describe exploring neural inspired approaches as a novel alternative method.

Chapter 2

Approach

As many of the canonical computer science approaches to resource allocation have NP-Hard complexity, rather in this research effort we seek to explore whether neural inspired computation can provide an advantageous approach for high performance computing. The brain intrinsically balances computation and encoding while incorporating different time scales, memory capacities, and forms of computation. In particular, neurogenesis, the addition and inclusion of neurons in a functioning neural circuit is the neuroscience process we are examining in this case study.

As follows, we introduce the neurogenesis process, describe a variety of methods for how it may impact memory management, and then present a computational modeling and simulation paradigm to explore the potential impact.

2.1 Neurogenesis

The process of adult neurogenesis in the dentate gyrus (DG) region (shown in Fig. 2.1) is reviewed extensively in Aimone et al. [2], but we briefly survey it here. Each day, roughly 1,000 new neurons are born from a stem cell population that resides locally within the DG. This rate is highly regulated by a number of intrinsic and extrinsic factors, as is the ultimate survival of the neurons that are born. While numbers differ somewhat from study to study, within several weeks about half of the neurons that are born no longer exist, most likely due to activation of apoptotic pathways (an internal gene signaling cellular death mechanism). If a neuron lives to about four to six weeks old, it most likely will persist indefinitely.

In rodents, new neurons take approximately two months to achieve maturity. During this time, they progress from a neuroblast cell phenotype, which lacks the projections commonly associated with neurons, to fully functional granule cells that are indistinguishable from those born at earlier ages. Once new input and output synapses start to form at about 14 to 16 days old, the cells mature rapidly, obtaining new synapses at a rapid pace. By about two months old, the neurons have about 5,000 to 6,000 input glutamatergic synapses from both internal and external cortical populations. A final key observation is that synapses on young neurons are more plastic, i.e., more amenable to learning, than those on mature neurons.

In addition to this difference in connectivity and synaptic plasticity, young neurons are distinct

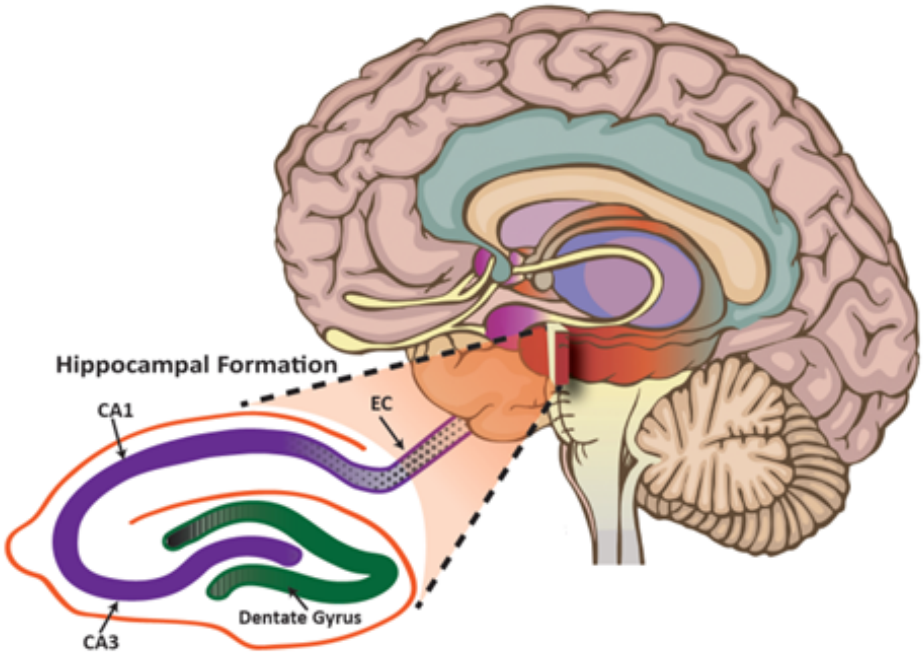


Figure 2.1. Dentate gyrus region (in green) of the hippocampus

from the mature neurons in their basic electrophysiological properties. Young neurons typically have a higher membrane resistance, which allows individual synapses to have a higher relative impact than a similar weighted (same maximum conductance) synapse on a mature neuron. The combination of these properties the physiology, connectivity, and plasticity of young neurons has led to a widespread acceptance that these cells are more active or hyperexcitable compared to the mature population [2].

The physiological implications of this maturation process provides evidence for a mixed coding role described extensively in [1]. As portrayed in Figure 2.2, the general idea is that mature neurons are tightly tuned to respond to specific stimuli, and conversely immature neurons are more broadly responsive and able to encode novelty. These response characteristics are portrayed by the blue and green tuning curves for the mature and immature neurons on parts B and A of the figure respectively. Existing together in the same system, the mixture of both mature and immature neurons in effect enables the system to encode known information while also being able to adapt and encompass novel previously unseen content.

2.2 Insights for Memory Management

There are a variety of ways in which the neurogenesis process can provide insights and potential advantages for MLM management. One approach is to relate different memory types to mature

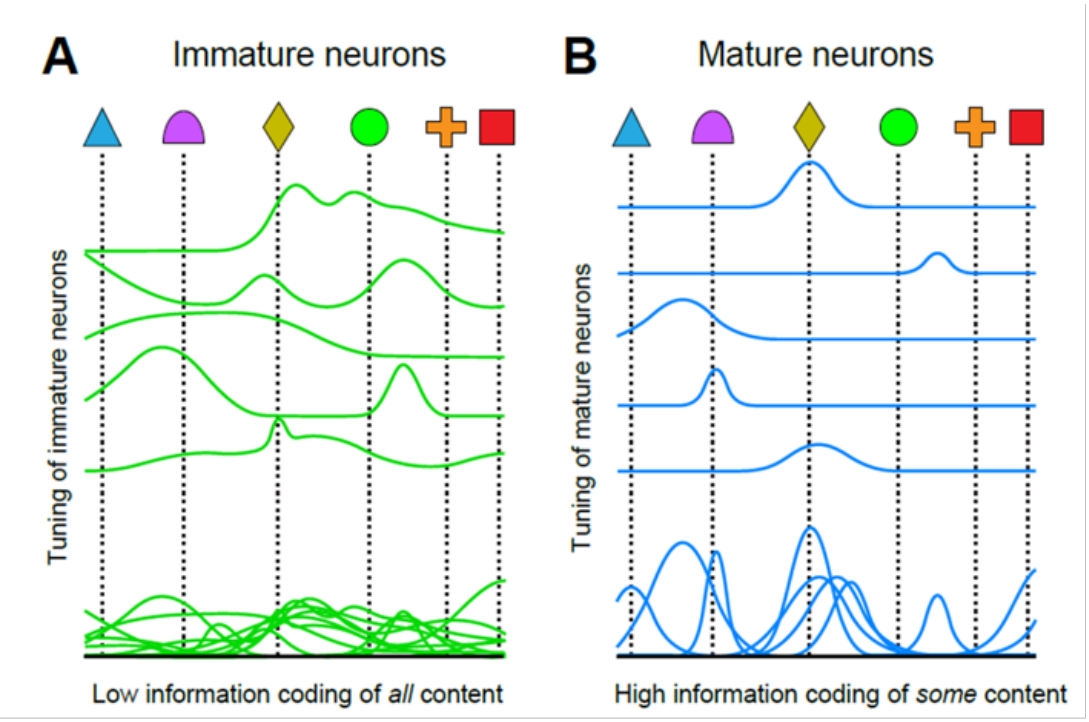


Figure 2.2. Mixed coding paradigm

and immature neurons. In this sense, just as neurons of differing levels of maturity have differing response characteristics, the different memory types of the MLM architecture have different characteristics as well. And so one possible mapping methodology is to correlate high bandwidth memory with mature (tightly tuned) neurons and conversely correlate lower bandwidth but high capacity memory with immature (broadly tuned) neurons. In this sense, neuron maturity is related to how readily the variables stored in memory are needed by the program. Conversely, since immature neurons are believed to encode novelty, the sense of newness and recency associated with novelty suggests a mapping where the immature neurons correspond to the low bandwidth proximal memory technologies. Regardless of the mapping, just as there are a spectrum of neuron maturity levels, these mapping paradigms can be distributed across however many levels of memory are in the MLM architecture.

Alternatively, storage within a given memory type could be allocated analogously to the mixed coding paradigm of neurogenesis. In this sense, a notion of maturation would reserve a percentage of memory addresses for program variables meeting that maturation level while simultaneously reserving some of the memory addresses for less established variables that may meet an emerging need as the program being computed progresses.

In addition to relating the memory levels and their corresponding characteristics of a MLM

architecture to neurogenesis, algorithmic approaches may also be taken. Similar to the notion of inter-memory type address allocation inspired by neurogenesis, a similar approach may be taken at the algorithmic level in developing a neurogenesis inspired data structure or encoding. Rather than reserving the most efficient representations solely for the most likely data as Huffman encoding does, a percentage of the top representations could be reserved for new concepts. Such an approach has the potential to yield an adaptive representation.

Another algorithmic approach would be to develop the memory controller logic to operate inspired by principles of the neurogensis process. This approach would operate upon program flow heuristics to assign maturity levels to the program variables and use that information to decide what variables should be added and removed from each memory level across time.

However, in the remainder of this manuscript we will focus upon and present results from exploring mixtures of memory controllers analogous to a mixed coding.

2.3 Computational Model

To explore the impact of mixtures of memory controllers, we developed a computational simulation model as shown by Figure 2.3.

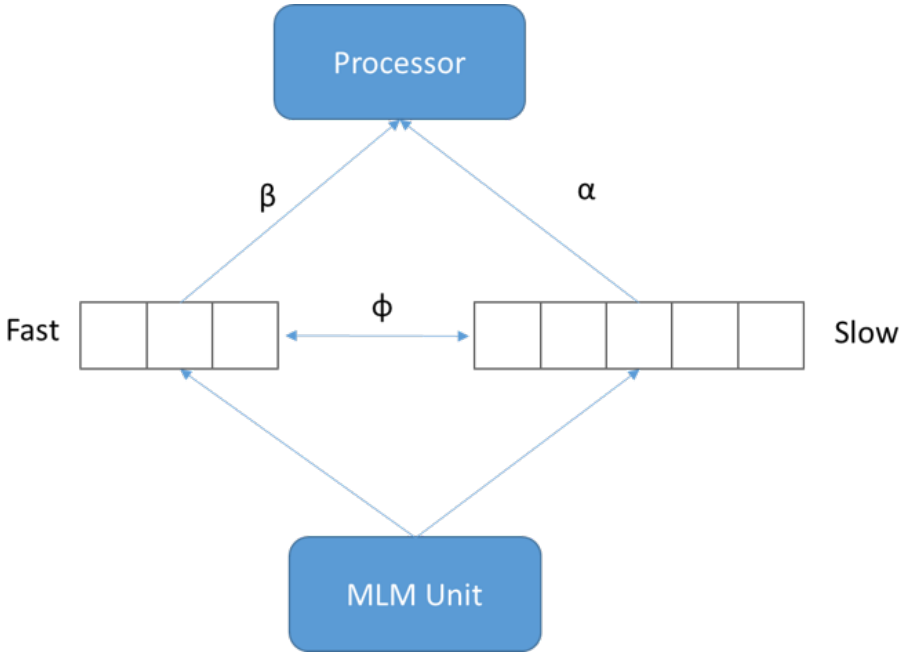


Figure 2.3. Two level computational simulation network

In the model represented by the figure, we explored the role of two levels of post cache memory labeled as fast and slow in the illustration. The ”fast” memory represents a lower capacity higher

bandwidth memory type and the ”slow” memory represents a higher capacity lower bandwidth memory type. While only two memory levels are captured here, larger MLM architectures could also be examined. Each of these memories has finite sizes (denoted by the number of boxes) as well as access costs associated with transmitting data from the memory types and swapping variables between memory. In addition to the architectural parameters, the program being executed on the MLM architecture also imposes additional parameters which impact operation. In particular, the program length, program flow, and number of unique variables all impact the operation. For example, if the number of unique variables in a program of interest were smaller than the capacity of fast memory the trivially optimal memory allocation would be to place everything in the fastest memory. Next we describe simulation results exploring this model space.

Chapter 3

Results

The particular program being executed on a high performance computing architecture can result in significant performance variability. For this case study exploration we focused upon the memory distributions yielded by a few scientific mini-apps. These mini-apps are program representations which encapsulate the key computations of a larger scientific application with the intention of enabling benchmarking without necessitating the full large program be run [9].

In particular, the four primary mini-apps focused upon are: miniFE, rsbench, lulesh, and miniAero. Briefly, miniFE is a mini-app that mimics the finite element generation, assembly and solution for an unstructured grid problem. Part A of Figure 3.3 illustrates the memory access histogram of the miniFE mini-app. As shown, this mini-app has a wide number of variables with fairly even accesses. Rsbench is a molecular dynamics mini-app with considerable variation in the memory access. The memory access histogram for rsbench is shown in part A of Figure 3.4. Lulesh is a a hydrodynamics application exhibiting a very unequal distribution of memory accesses. Less than 5% of pages in main memory account for more than half of the memory accesses and an additional 15% of pages account for the vast bulk of memory requests. This variability is shown in part A of Figure 3.5. And lastly, miniAero is an aerodynamics mini-app exhibiting a few well-defined variables. The miniAero memory access histogram is illustrated in Figure 3.6. For more detail about these mini-apps see [9]. In the computational simulation experiments explored in this project, these access histograms were used to generate program distributions of varying lengths and program flow.

To explore mixtures of memory controllers, we used four common addition controller policies and four replacement controller policies. Addition policies decide whether or not a variable should be added to fast memory. The four addition policies and a brief description of their logic are as follows:

- addMFU (Most Frequently Used): this policy uses a count of how many times each memory variable has been used and if the candidate memory variable's count exceeds the min of all variables in fast memory it is added
- addRand: this policy randomly decides (via a coin toss) whether the candidate memory variable will be added. Note there are other randomization approaches which also may be implemented, such as using an n-sided coin toss with n related to memory sizes.
- addMRPU (More Recent Previous Use): this policy uses a history timestamp of each vari-

ables last use and if the candidate memory variables last use is more recent than the least previously used variable in memory it is added.

- addMFPRU (More Frequent, More Recent Previous Use): this policy requires that a candidate variable must have a greater frequency usage than the least recently used variable to be added.

Replacement policies decide what should be replaced within fast memory. The four replacement policies and a brief description of their logic are as follows:

- FIFO (First In/First Out): this policy maintains a list of when variables were added to memory and when a new variable needs to be added the variable at the front of the list is removed and the new variable is added to the back of the list.
- LRU (Least Recently Used): this policy uses an ordered list of usage time-stamps and removes the variables used least recently
- LFU (Least Frequently Used): this policy uses the frequency counts of the variables in fast memory and removes the variable with the lowest usage.
- Rand: this policy randomly selects a variable from fast memory to remove.

To decide whether to add a variable to memory and if so what to remove, both an addition and replacement policy must be coupled to combined yield a memory controller policy. Using the addition and replacement policies described above yields 16 addition-replacement policies. Other addition or replacement policies could also be considered, such as a novel policy implementing neurogenesis inspired logic as described in Section 2.2. For further details of these and other policies see [7].

3.1 Analysis of Strategies

To measure the efficiency of memory control policies, while it is desirable to send many variables from fast memory, the number of fast memory transfers alone is not a sufficient metric as this could be maximized by routing all variable calls through fast memory. However, there is a cost in swapping variables in memory, and so the number of swaps must also be accounted for. Thus, by measuring the number of fast memory transfers minus swaps strives to identify controller policies which most efficiently place variables in memory.

Examining the fast transfers minus swaps efficiency measure of single memory control policy over time identifies how well the controllers are suited to different memory access distributions elicited by the mini-apps as well as temporal effects to their usage. Figure 3.1 and Figure 3.2 illustrate scaled distributions for miniFE and rsbench. In these plots, it may be seen that no single control policy is ideally suited irrespective of the program domain. And additionally, by looking

at the policies across time, it may be observed that even within a single mini-app a single policy is not optimal uniformly across time. Rather, there are temporal intervals where different policies are locally the most efficient.

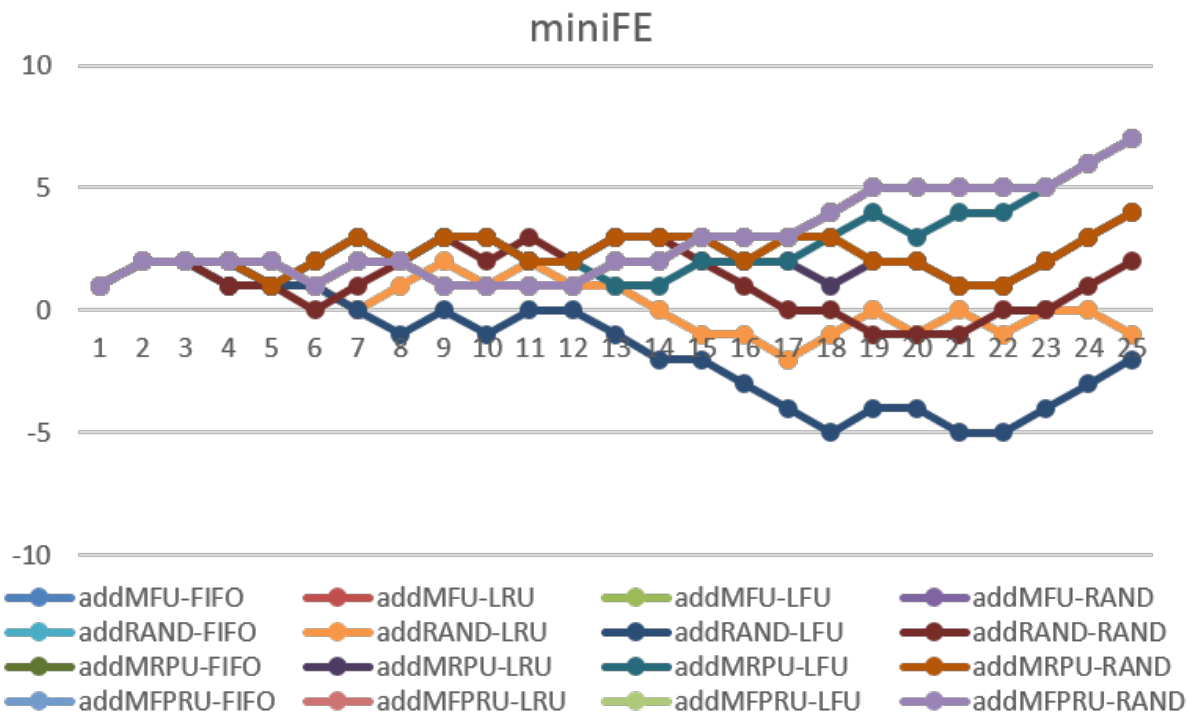


Figure 3.1. miniFE MiniApp Results

3.2 Impact of Mixed Strategies

Instead of relying upon a single memory control policy, rather, we explored neurogenesis inspired mixed control policy strategies. In this sense, just as a mixed coding combines mature and immature neurons to allow for adaptivity, a mixed memory control strategy employs different policies over time striving for a more efficient memory allocation.

There are exponentially many ways in which memory control policies may be combined. For this case study, we have considered some two policy combinations. Part B of Figure 3.3, Figure 3.4, Figure 3.5, and Figure 3.6 each illustrate averaged results of employing two control policies on a given mini-app. The orange columns provide the baseline of how a single controller performs, and the blue column combines that single controller with another controller. In the examples shown, the second controller is the MFU-LFU policy. These results are attained using a 20-80 mix (this ratio being inspired by neurogenesis ratios) where the first control policy is run for the first 20%

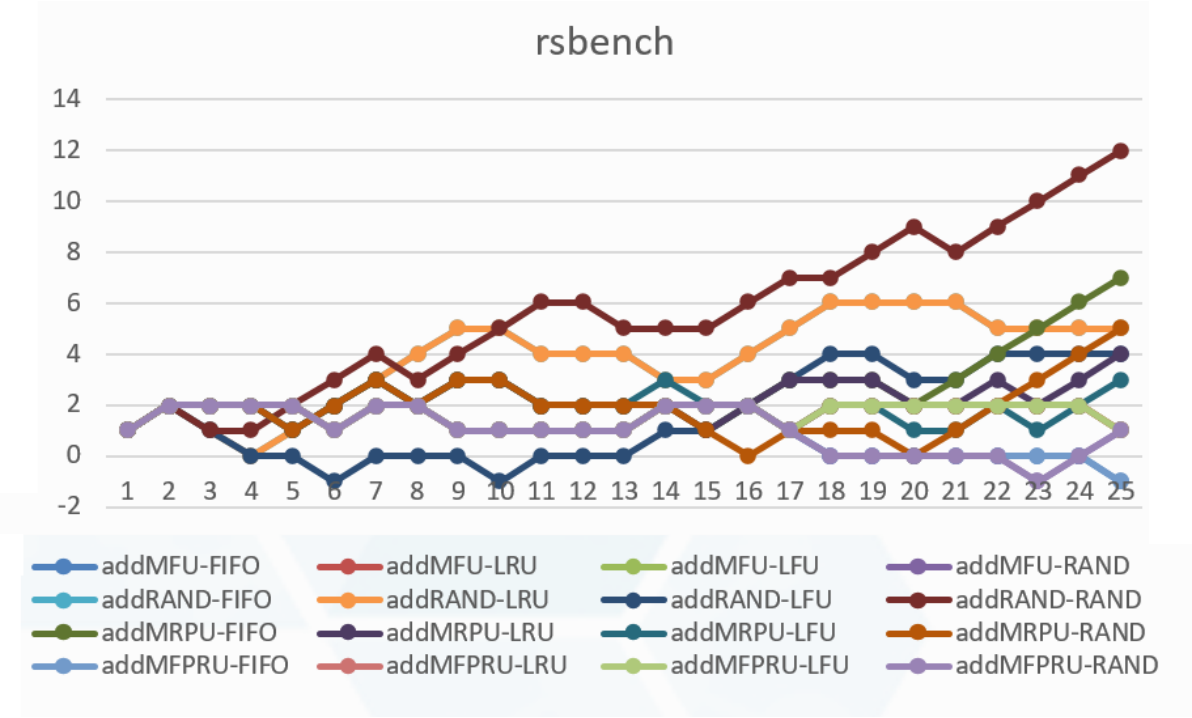


Figure 3.2. rsbench MiniApp Results

of the program duration and the second control policy for the remaining 80%. As shown, the blue (mixed policies) are either as good or better than the single policy control establishing a meaningful benefit to a neural inspired mixed control policy across all the mini-apps considered here.

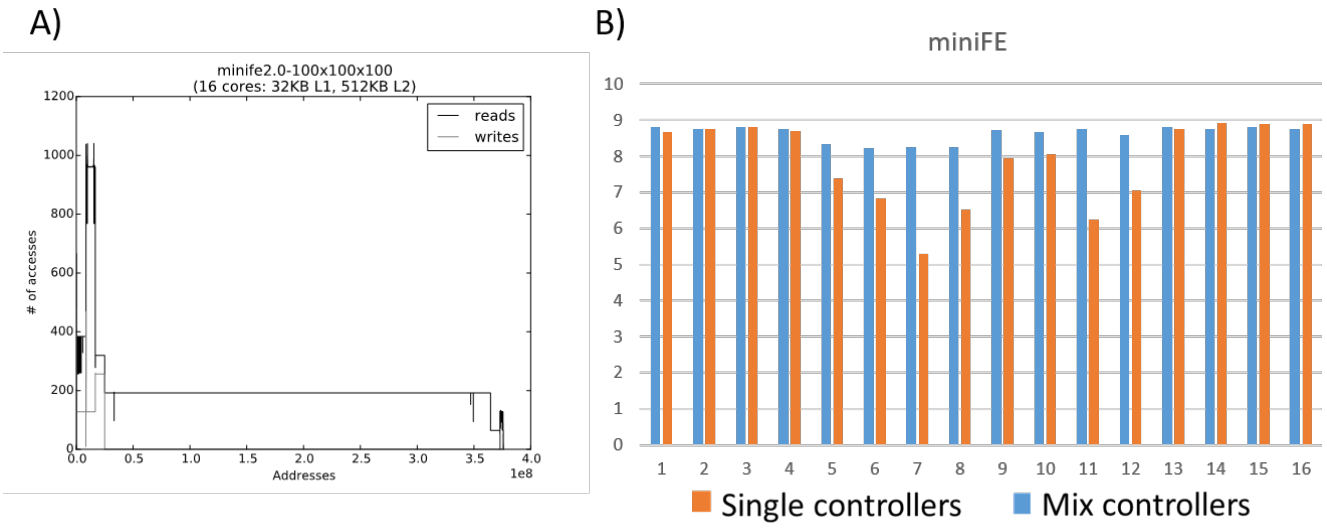


Figure 3.3. miniFE MiniApp Results

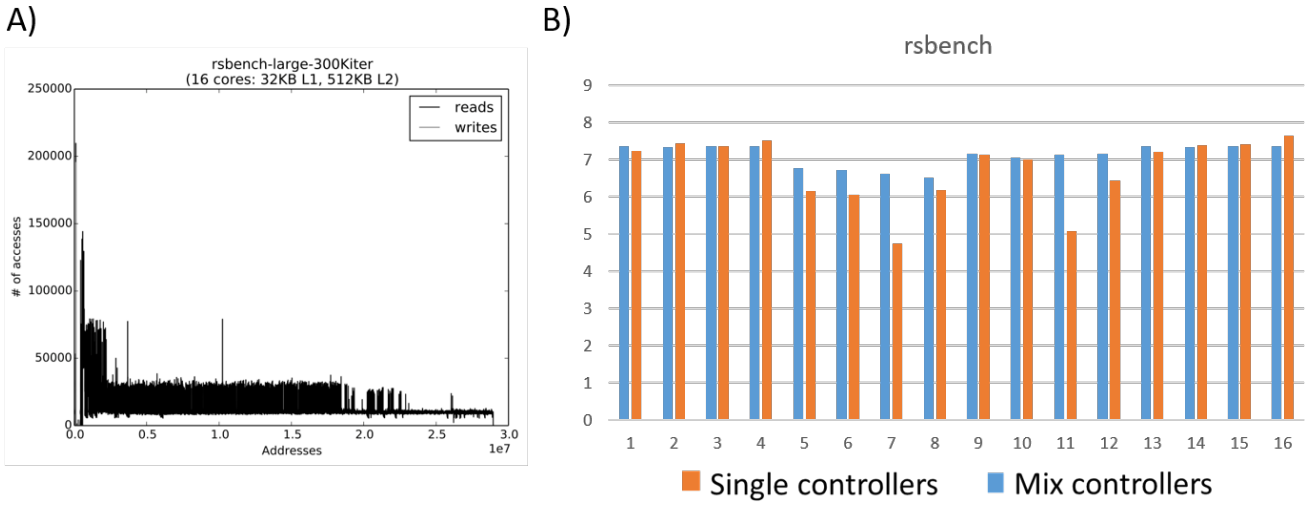


Figure 3.4. rsbench MiniApp Results

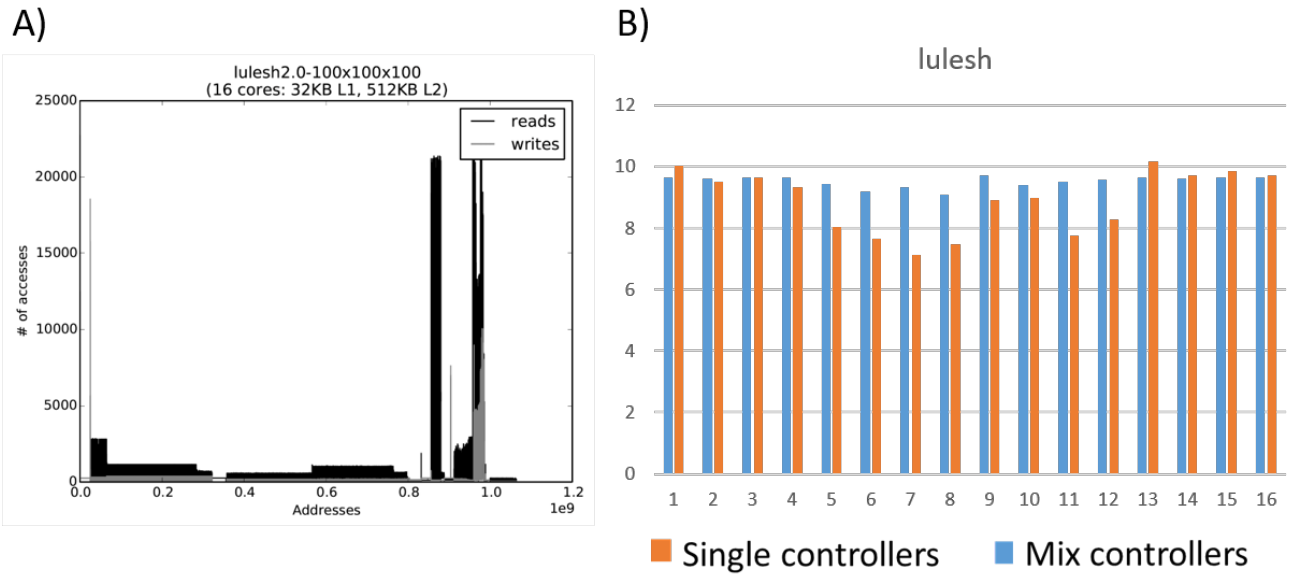


Figure 3.5. lulesh MiniApp Results

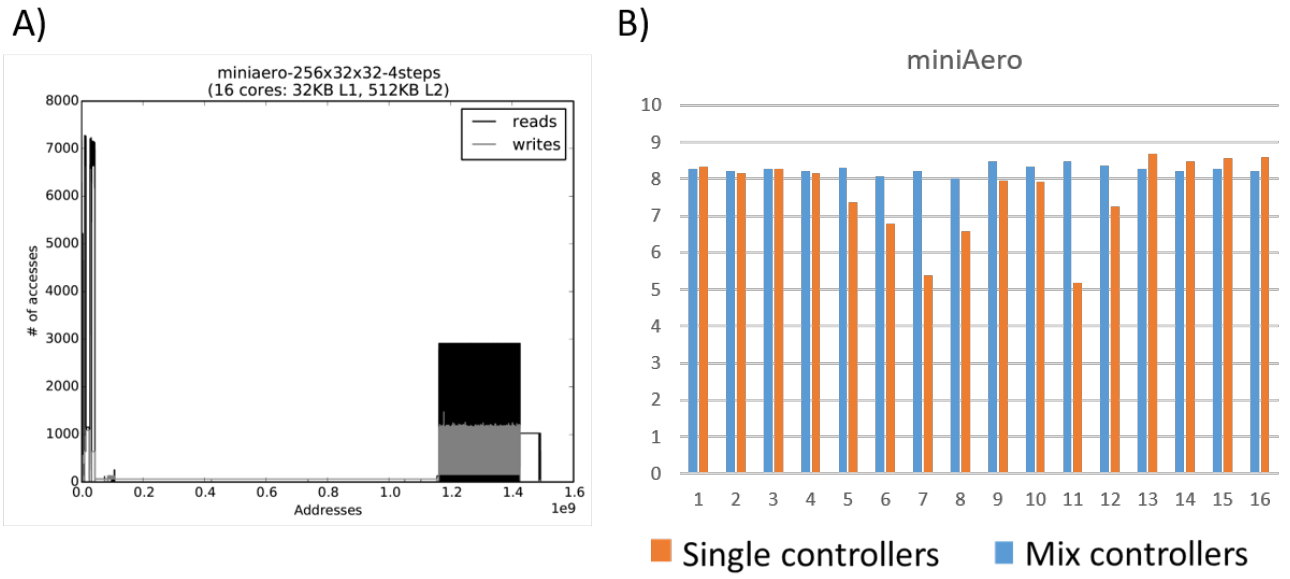


Figure 3.6. miniAero MiniApp Results

Chapter 4

Conclusion

This case study on neural inspired dynamic memory management strategies for high performance computing successfully established that neural-inspired approaches can beneficially impact how emerging MLM architectures utilize memory. In doing so, it provides a basis for a more fundamental understanding of the interplay between architectures and programs. Rather than simply employing the most empirically successful memory control policy, a mathematical understanding may be further pursued. For example, in the mathematics of game theory, the implications of factors such as the number of game iterations or the characteristics of an opponent can yield insights into the conflict being modeled. Likewise, by further exploring the parameter space and MLM architecture interactions fundamental understandings of memory control policies may be possible. Doing so would not only allow for the development of more efficient resource utilization, but would also provide insights into how to design more efficient architectures with specific computational goals in mind.

This successful case study exploration also provides motivation for more advanced reinforcement learning approaches to learn sophisticated strategies. Rather than manually exploring mixed strategies of control policy combinations, reinforcement learning is a principled learning algorithm which is able to learn sophisticated strategies given a performance feedback measure. In addition to pursing the more sophisticated learning approach, a formal computational characterization will also provide insight into how quickly a policy algorithm may adapt as well as how computationally intensive an adaptive learning policy is. Such characterizations can impact at what level neural inspired resource management approaches may be employed.

4.1 Future Work

In addition to the future directions mentioned above, other efforts would be to further explore the additional potential neurogenesis MLM mappings mentioned in Section 2.2. Adaptive algorithm and data structure advances are a likely outcome in pursuing these alternative approaches. Additionally, with advances being made in neuromorphic computing hardware, that is computational devices which mimic the functionality and operation of the brain, there is a opportunity to explore how these devices may impact high performance computing. Neuromorphic devices could serve in an accelerator role complementing the computational role of HPC. Or additionally, they could provide the implementation medium of neural inspired control mechanisms. And finally, other

HPC operations can likewise be explored for potential neural-inspired advantages beyond simply the MLM resource management case study examined in this research effort.

References

[1] James B Aimone, Wei Deng, and Fred H Gage. Resolving new memories: a critical look at the dentate gyrus, adult neurogenesis, and pattern separation. *Neuron*, 70(4):589–596, 2011.

[2] James B Aimone, Yan Li, Star W Lee, Gregory D Clemenson, Wei Deng, and Fred H Gage. Regulation and function of adult neurogenesis: from genes to cognition. *Physiological reviews*, 94(4):991–1026, 2014.

[3] Joseph Altman and Gopal D Das. Autoradiographic and histological evidence of postnatal hippocampal neurogenesis in rats. *The Journal of comparative neurology*, 124(3):319–335, 1965.

[4] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction to algorithms second edition, 2001.

[5] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[6] Charles G Gross. Neurogenesis in the adult brain: death of a dogma. *Nature Reviews Neuroscience*, 1(1):67–73, 2000.

[7] Simon D Hammond, Arun F Rodrigues, and Gwendolyn R Voskuilen. Multi-level memory policies: What you add is more important than what you take out. In *Proceedings of the Second International Symposium on Memory Systems*, pages 88–93. ACM, 2016.

[8] Darrell A Henze, Lucia Wittner, and György Buzsáki. Single granule cells reliably discharge targets in the hippocampal ca3 network in vivo. *Nature neuroscience*, 5(8):790–795, 2002.

[9] Michael A Heroux, Douglas W Doerfler, Paul S Crozier, James M Willenbring, H Carter Edwards, Alan Williams, Mahesh Rajan, Eric R Keiter, Heidi K Thornquist, and Robert W Numrich. Improving performance via mini-applications. *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, 3, 2009.

[10] Michael S Kaplan. Environment complexity stimulates visual cortex neurogenesis: death of a dogma and a research career. *Trends in neurosciences*, 24(10):617–620, 2001.

[11] Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *Information Theory, IEEE Transactions on*, 22(1):75–81, 1976.

[12] Bruce L McNaughton and Richard GM Morris. Hippocampal synaptic enhancement and information storage within a distributed memory system. *Trends in neurosciences*, 10(10):408–415, 1987.

[13] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.

[14] Randall C O'reilly and James L McClelland. Hippocampal conjunctive encoding, storage, and recall: avoiding a trade-off. *Hippocampus*, 4(6):661–682, 1994.

[15] Amar Sahay, Donald A Wilson, and René Hen. Pattern separation: a common function for new neurons in hippocampus and olfactory bulb. *Neuron*, 70(4):582–588, 2011.

[16] Adam Santoro and Michael Yassa. Pattern separation disambiguating the similar since 1971, April 2015.

[17] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[18] Kirsty L Spalding, Olaf Bergmann, Kanar Alkass, Samuel Bernard, Mehran Salehpour, Hagen B Huttner, Emil Boström, Isabelle Westerlund, Céline Vial, Bruce A Buchholz, et al. Dynamics of hippocampal neurogenesis in adult humans. *Cell*, 153(6):1219–1227, 2013.

[19] Janusz Szczepanski, José M Amigó, E Wajnryb, and MV Sanchez-Vives. Application of lempel-ziv complexity to the analysis of neural discharges. *Network: Computation in Neural Systems*, 14(2):335–350, 2003.

[20] Alessandro Treves and Edmund T Rolls. Computational constraints suggest the need for two distinct input systems to the hippocampal ca3 network. *Hippocampus*, 2(2):189–199, 1992.

[21] Jonathan D Victor. Approaches to information-theoretic analysis of neural activity. *Biological theory*, 1(3):302, 2006.

[22] Jonathan D Victor. Approaches to information-theoretic analysis of neural activity. *Biological theory*, 1(3):302–316, 2006.

[23] Craig M Vineyard, Stephen J Verzi, Thomas P Caudell, Michael L Bernard, and James B Aimone. Adult neurogenesis: Implications on human and computational decision making. In *Foundations of Augmented Cognition*, pages 531–540. Springer, 2013.

[24] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

[25] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on*, 24(5):530–536, 1978.

Appendix A

Publications

Included in this appendix are the abstracts and reference citations of the publications associated with this project. These publications include topics such as understanding neurogensis encodings, adaptive algorithms, resource allocation models, and neural optimization approaches. For more information see the full papers.

A.1 Quantifying Neural Information Content: A Case Study of the Impact of Hippocampal Adult Neurogenesis

Through various means of structural and synaptic plasticity enabling online learning, neural networks are constantly reconfiguring their computational functionality. Neural information content is embodied within the configurations, representations, and computations of neural networks. To explore neural information content, we have developed metrics and computational paradigms to quantify neural information content. We have observed that conventional compression methods may help overcome some of the limiting factors of standard information theoretic techniques employed in neuroscience, and allows us to approximate information in neural data. To do so we have used compressibility as a measure of complexity in order to estimate entropy to quantitatively assess information content of neural ensembles. Using Lempel-Ziv compression we are able to assess the rate of generation of new patterns across a neural ensembles firing activity over time to approximate the information content encoded by a neural circuit. As a specific case study, we have been investigating the effect of neural mixed coding schemes due to hippocampal adult neurogenesis.

Vineyard, C.M., Verzi, S.J., James, C.D., & Aimone, J.B. (2016) Quantifying Neural Information Content: A Case Study of the Impact of Hippocampal Adult Neurogenesis (IJCNN 2016)

A.2 Overcoming the Static Learning Bottleneck - the Need for Adaptive Neural Learning

Amidst the rising impact of machine learning and the popularity of deep neural networks, learning theory is not a solved problem. With the emergence of neuromorphic computing as a means of addressing the von Neumann bottleneck, it is not simply a matter of employing existing algorithms on new hardware technology, but rather richer theory is needed to guide advances. In particular, there is a need for a richer understanding of the role of adaptivity in neural learning to provide a foundation upon which architectures and devices may be built. Modern machine learning algorithms lack adaptive learning, in that they are dominated by a costly training phase after which they no longer learn. The brain on the other hand is continuously learning and provides a basis for which new mathematical theories may be developed to greatly enrich the computational capabilities of learning systems. Game theory provides one alternative mathematical perspective analyzing strategic interactions and as such is well suited to learning theory.

Vineyard, C.M., & Verzi, S.J. (2016) Overcoming the Static Learning Bottleneck - the Need for Adaptive Neural Learning (ICRC 2016)

A.3 FlipIt is a Game of Chicken

FlipIt is an abstract cyber-security game designed to in- vestigate optimal strategies for managing security resources in response to Advanced Persistent Threats. In this paper, we establish bounds on the optimal benefits and rates of play for FlipIt, and show that the best strategy for many variants of the game involves presenting a credible threat to potential players. Because of this, FlipIt is most like the game of chicken. We apply our results to the analysis of Advanced Persistent Threats and discuss the value of FlipIt to cyber security research.

Hobbs, J. & Benson, J. (2016) FlipIt is a Game of Chicken. In Algorithmic Game Theory workshop at IJCAI 2016

A.4 Computing with Spikes: The Advantage of Finegrained Timing

Neural-inspired spike-based computing machines often claim to achieve considerable advantages in terms of energy and time efficiency by using spikes for computation and communication; however, fundamental questions about spike-based computation remain unanswered. For instance, how much advantage do spike-based approaches have over conventional methods and under what circumstances does spike-based computing provide a comparative advantage? Simply implementing existing algorithms using spikes as the medium of computation and communication is not

guaranteed to yield an advantage. Here, we demonstrate that spike-based communication and computation within algorithms can increase throughput and in some cases decrease energy cost. We present several fundamental spiking algorithms, including sorting a set of numbers in ascending/descending order as well as finding the maximum/minimum or median of a set of numbers. A formal trade-space analysis of these algorithms with respect to important performance metrics such as runtime, number of processors and energy consumption, will allow us to optimize operating conditions of these algorithms with respect to specific neural architectures (e.g., BrainScaleS, Neurogrid, SpiNNaker, TrueNorth, the Spike-Timing Processing Unit, etc.). In addition an example application is provided, a spiking median filtering approach for image processing providing a low energy, parallel implementation.

Verzi, S.J., Rothganger, F., Parekh, O.D., Quach, T., Miner, N.E., Vineyard, C.M., James, C.D., Aimone, J.B. (in preparation) Computing with Spikes: The Advantage of Fine-grained Timing (Neural Computation Journal)

Appendix B

Project Summary Poster

As shown on the following page is the project summary poster presented at the 2017 Computing and Information Science (CIS) External Review Board (ERB).

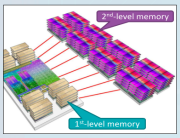
Neural-Inspired Algorithms for HPC
a Memory Management Case Study

Sandia National Laboratories
C.M. Vineyard, J.B. Aimone, M. Galiardi, and S.J. Verzi
Sandia National Laboratories, New Mexico

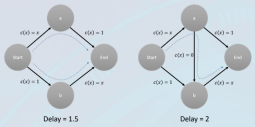
Problem

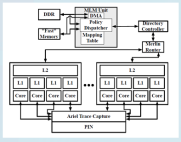
Exascale-class supercomputers will require unprecedented amounts of memory bandwidth and capacity - To satisfy these requirements, vendors are proposing Multi-Level Memory (MLM) combining different memory technologies in a single system

- **Multi-Level Memory (MLM)** – mash-up (of different technologies)
  - Main memory is comprised of two or more types of memory instead of conventional single-level DDR DRAM-only main memory
  - High-bandwidth memories to serve majority of accesses
  - Low cost memory to provide majority of capacity



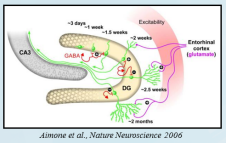
- Introduces the challenge of - How scientific computing apps make use of this capability
  - Braess' Paradox: increased capacity not guaranteed to be advantageous





Approach

Take inspiration from neuroscience to explore the ability of neural inspired dynamic memory management strategies to adapt to different applications on different HPC architectures



Aimone et al., Nature Neuroscience 2006

Adult Neurogenesis

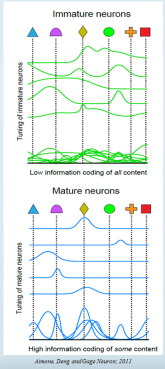
- The incorporation of new neurons into established, functioning neural circuits
- Extended maturation
  - Several weeks to begin integrating into circuit
  - Still "immature" several months later

Mixed Coding Hypothesis

- Neurogenesis provides mixed coding scheme for Dentate Gyrus
- Mature neurons are tightly tuned and sparsely active – they respond to few things very strongly
- Young neurons are more broadly tuned and more active – they respond to more things less strongly

Applied to Memory Management

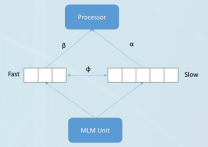
- Several possible approaches:
  - Neural-inspired: data structures, memory controller logic, neuromorphic hardware
- Mixtures of memory controllers analogous to mixed coding



Aimone, Deng and Gage Neuron, 2011

Results

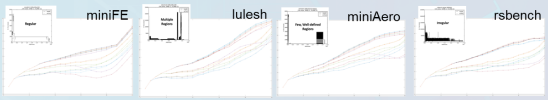
Computational modeling of post cache memory calls for a 2 level architecture



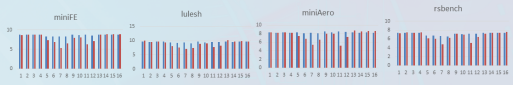
Model Parameters:

- Memory sizes
- Access costs
- Program variability
  - Length
  - Flow
  - Number of unique variables

Simulation Results of Fast Memory Accesses Minus Swaps for Different Controllers over Time



20-80 Controller Mix (MFU-LFU as Second Controller)

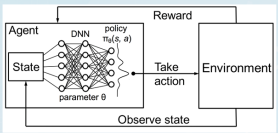


Single controllers Mix controllers

Illustration of Variability of Controllers

Significance

- Establishes that neural-inspired approaches can beneficially impact how emerging MLM architectures utilize memory
- Provides basis for more fundamental understanding of interplay between architectures and programs
- Motivation for more advanced reinforcement learning approaches to learn sophisticated strategies



- Other HPC operations can likewise be explored for potential neural-inspired advantages

U.S. DEPARTMENT OF ENERGY
National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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.

SAND 2017-8618D

Sandia National Laboratories

Figure B.1. CIS 2017 ERB Poster

DISTRIBUTION:

1	MS 0899	D. Chavez, LDRD Office, 1911
1	MS 0899	Technical Library, 9536 (electronic copy)


Sandia National Laboratories