ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Lemont, Illinois 60439

Overcoming the Power Wall by Exploiting Application Inexactness and Emerging COTS Architectural Features[1]

Mike Fagan, Jeremy Schlachter, Kazutomo Yoshii, Sven Leffyer, Krishna Palem, Marc Snir, Stefan M. Wild, and Christian Enz

Mathematics and Computer Science Division

Preprint ANL/MCS-P6040-0816

August 2016

Updates to this preprint may be found at
http://www.mcs.anl.gov/publications

[1]This material was based upon work supported by the U.S. Department of Energy, Office of Science, Offices of Basic Energy Sciences and Advanced Scientific Computing Research under Contract No. DE-AC02-06CH11357.

Overcoming the Power Wall by Exploiting Application Inexactness and Emerging COTS Architectural Features

Trading Precision for Improving Application Quality

Mike Fagan[†], Jeremy Schlachter[‡], Kazutomo Yoshii[*],
Sven Leffyer[*], Krishna Palem[†], Marc Snir[*], Stefan M. Wild[*], and Christian Enz[‡]

[*]Mathematics and Computer Science Division
Argonne National Laboratory, Lemont, IL 60439, USA

[†]Department of Computer Science
Rice University, Houston, TX, USA

[‡]Integrated Circuits Laboratory (ICLAB)
Ecole Polytechnique Fédérale de Lausanne (EPFL), Neuchâtel, Switzerland

***Abstract*—Energy and power consumption are major limitations to continued scaling of computing systems. Inexactness where the quality of the solution can be traded for energy savings has been proposed as a counterintuitive approach to overcoming those limitation. However, in the past, inexactness has been necessitated the need for highly customized or specialized hardware. In order to move away from customization, in earlier work [4], it was shown that by interpreting precision in the computation to be the parameter to trade to achieve inexactness, weather prediction and page rank could both benefit in terms of yielding energy savings through reduced precision, while *preserving* the quality of the application. However, this required representations of numbers that were not readily available on *commercial off-the-shelf* (COTS) processors. In this paper, we provide opportunities for extending the the notion of trading precision for energy savings into the world COTS. We provide a model and analyze the opportunities and behavior of all three IEEE compliant precision values available on COTS processors: (i) double (ii) single, and (iii) half. Through measurements, we show through a limit study energy savings in going from double to half precision can potentially exceed a factor of four, largely due to memory and cache effects.**

I. Introduction

It is widely believed that *energy* and *power* consumption are major limitations to continued scaling of computing systems. Often referred to as the "power wall" (or "energy wall"), this limitation now ranges from the obvious battery-constrained context of embedded computing, to general-purpose computing settings, namely supercomputers and data-centers.

Following the presidential executive order on creating a national strategic computing initiative [1], US agencies are engaged in a project aimed at leading to the deployment of an exascale computer by 2023. Power consumption is a major obstacle to the timely deployment of an exascale platform. The current top supercomputer, Sunway, consumes 15.3MW and achieves 93 petaflop/s on Linpack. With no technology improvements, an increase in performance would require a proportional increase in power consumption: An exascale system would consume more than 160MW. New technology improvements will occur, but it seems unlikely that an exascale system will achieve the target of 20MW.

There is a lot of interest in ameliorating these hurdles through customization—notably through highly stylized and dedicated architectures. These efforts at customization are ubiquitous in the embedded computing space where energy, speed, and size/weight have always played a critical role. The solution in this traditional context is to consider *system-on-a-chip* (SOC) architectures that meld varying amounts of general-purpose embedded computing with custom processing hardware and communication hardware. The problem with SOC architectures is that customization comes with a significant "one-time" *nonrecurring engineering cost* (NRE and *time-to-market* delays [2]. On the other hand, with general-purpose *commercial, off-the-shelf* (COTS) microprocessors, a single highly optimized design is used to amortize the NRE and time-to-market overheads through volume sales.

In addition, customized hardware requires customized software: A specialized system is generally harder to program than a general purpose microprocessor and does not take as much advantage of the large software ecosystem that is available for general purpose processors. For example, Sunway uses a specialized SOC with no caches, but only a small 64K scratchpad for each core. This requires significant changes in software.

A general technique for reducing energy/power consumption goes by the name of *inexact design*. The philosophy of inexact design espouses the idea of actively trading application accuracy

This material is based upon work supported by the Swiss National Science Foundation project IneSoC under grant N° 200021-144418 and by the US Dept. of Energy, Office of Science, ASCR, under contract DE-AC02-06CH11357.

for disproportionately large energy gains; see [3] for an overview. Recently, this approach was considered in the context of two ubiquitous applications drawn from the weather and climate modeling domains through the IGCM model, and in the context of "big data" through the *PageRank* algorithm [4]. This work established the following thesis:

> For applications that occur quite naturally, e.g. IGCM, PageRank), trading the precision at which the algorithms are implemented garners energy savings without compromising the quality of the application.

The thesis specializes the inexact design methodology by interpreting inexactness as lowering the precision of the computation. This application of inexact design opens the door to an entirely new approach for coping with the power- or energy-wall facing computing.

The work from [4], however, needed architectures where the floating-point numbers did not conform to IEEE standards. Therefore, while being a harbinger of how one could leverage COTS platforms to overcome energy hurdles, the methods could not be supported on currently-available microprocessors. Nevertheless, the insight from [4] is potentially useful if we apply the same methodology in the context of commercially available precision variants. This goal paves the way to the following questions addressed in this paper.

1) What are the most frequently occurring precision modes in COTS microprocessors?
2) What are the costs of various (e.g., integer, floating-point, load, store) instruction types as we vary precision?
3) What are realistic limits to *measured* energy gains we can hope to achieve across these classes?
4) What is a good model through which these measured gains can be explained and the architectural contributions from elements such as data movement from main memory, various levels of cache, and the data-path inferred?

A. Overview of the rest of the paper

The IEEE standards have provided single- and double-precision floating-point for some time and, more recently, a half-precision mode supporting 5 exponent bits, 10 mantissa bits, and a sign bit. We consider all three of these modes here. To start with, in Sec. II, we outline a methodology through which we can reliably measure energy consumption based on novel hardware counters. Section III describes the three modes of arithmetic under consideration. In Sec. IV, we construct (by hand) a family of three *microbenchmarks* to exercise each of the three modes in turn and measure the energy consumed. We show that by lowering precision, we obtain gains that are achievable with current commercially available microprocessors. We based this study on an Intel core i7 4770 (3.40 GHz) processor. We are also interested, however, in being able to project the benefits of reduced precision in the context of other architectures as well. To do this, in Sec. V, we consider energy to be relative or normalized by the cheapest operation and use the dimensionless quantity *virtual joules*. We use this model to reason about the gains reported in Sec. IV and to provide a tool for being able to estimate energy savings if measurement frameworks for physical are not readily available.

B. Related work

Early work on trading the quality or accuracy of a computation for energy savings based on physical mechanisms including CMOS devices can be found in [5]–[7]. A historical perspective on this approach to energy savings with a partial survey of the field can be found in [8]. In building the model used in this paper, we use the results from [9] both to validate our work as well as in determining costs of individual instruction classes. The concept of trading precision for performance is well known in the supercomputing community [10, 11]. The novelty of our approach is to apply this concept in the context of energy efficiency, both in terms of physical measurements, as well as validated models. The reference list associated with each of the papers we cite provides a more complete citation record.

II. Our Experimental Methodology

Being interested in energy effects means that we must have some way of measuring the energy consumption of programs. For the COTS used in this work, we employed a DELL precision T1700 workstation operated by CentOS 7 and equipped with an Intel core i7 4770 (3.40 GHz) and 16 GB of DDR3 RAM. The operating system was Linux, kernel 4.3. All CPU cores were set to the minimal P-State except for one. The distinguished core was set to Turbo Boost.

This Intel architecture supports Running Average Power Limit (RAPL) hardware counters, which work much like any other hardware performance counter: start counter, do work, stop counter. The difference between start and stop values measures the energy. The pseudocode for a RAPL measurement is shown in Algorithm 1. We employed RAPL to measure the energy per instruction (EPI) of several instruction classes. Our measurements for these instruction classes appear in Sec. V. In most of the cases, RAPL measurements are consistent with physical measurements, but this methodology tends to underestimate the cost of main memory access [12].

Algorithm 1 Generic RAPL energy measurement

```
Allocate and fill memory with random values
Initialize RAPL
for i = 1, i < N, i++ do
    Possibly increment memory pointer by more than a cache line size (if need to nullify cache)
    General Work        ▷ Could be inline assembly
end for
Read RAPL
```

III. Precision Modes available on COTS systems

Almost all floating-point units (FPUs) in commercially available processors support both IEEE 754 single-precision and double-precision arithmetic natively. To increase the peak performance, modern FPUs support single instruction, multiple data (SIMD) and fused multiply-add (FMA).

While single- and double-precision arithmetic are standard, there are many different half-precision formats available. There is, surprisingly, a standard half-precision format supported by IEEE-754-2008. This standard is named "binary16". It has

TABLE I: Energy for Fused Multiply-Add

Double-precision	199 J
Single-precision	67 J
Half-precision	49 J

5 bits of exponent, 10 bits of mantissa (11 if you count the leading 1-bit), and a sign bit. The IEEE standard, however, treats binary16 as a storage format *only*. General-purpose processors have, until recently, not supported binary16.

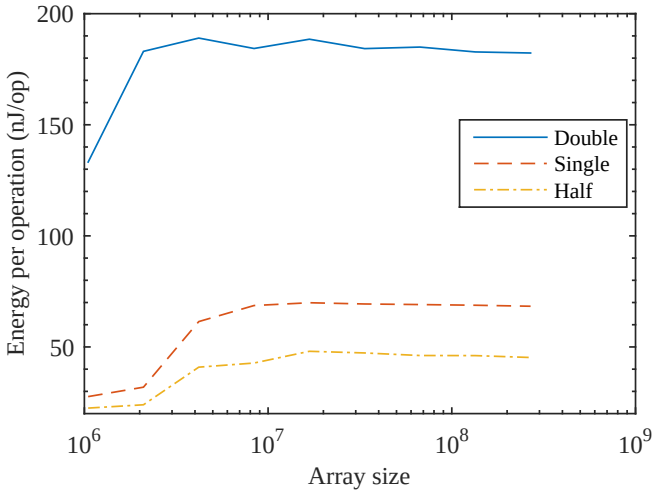
Intel recently introduced a half-precision floating-point storage format into the AVX2 instruction subset in the 3rd generation Intel Core processor family [13]. Specifically, Intel introduced instructions to convert half-precision data into single-precision data and convert single-precision data back into half-precision data. There is, unfortunately, no native support for half-precision arithmetic in AVX2. Any arithmetic to be done with binary16 data must be converted to standard single-precision to perform any calculations. At the end of the calculations, the (single-precision) results may be converted back to half precision.

Although actual floating-point arithmetic for half-precision data is carried out by single-precision logic, reducing data traffic size between processors and main memory simply benefits memory-bound applications since memory-wall-induced underutilization is still one of the major problems in scientific computing [14]. Even worse, Moore's law has raised the memory wall. In the exascale computing era, data movement is expected to be one of the most dominant factors for performance and energy efficiency [15].

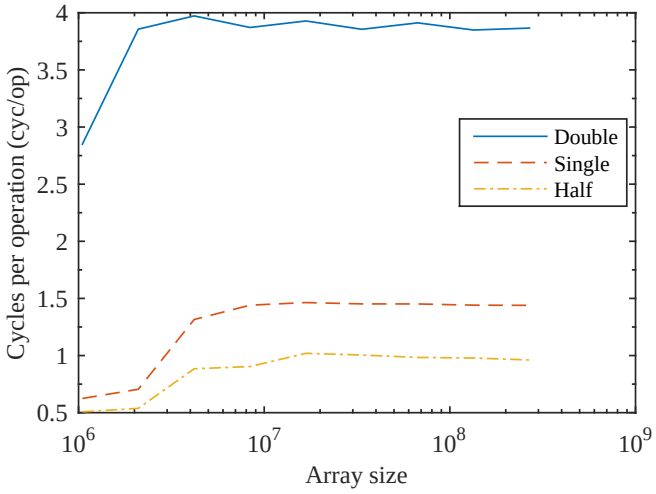
IV. Limit Study

This section gives a sample of how precision can be traded for a significant amount of energy saving on COTS (AVX2) hardware. To see the effects, we constructed a small benchmark consisting of 3 vector loads, 1 vector fused multiply-add, and 1 vector store. The entire sequence was looped over for a sufficient amount of time to be able to accurately measure the energy consumed. The benchmark has been run in 3 modes: double precision, single precision, and half precision.

Figure 1 shows the energy per operation, as it varies with array size. The asymptotic energy numbers appear in Table I.



(a)



(b)

Fig. 1: Energy per operation and cycles per operation of the vector fused multiply-add benchmark.

V. Modeling and Explaining the Gains

In order to understand the energy effects of various program designs, we must understand the source of the gain (or loss). The classic way to do that is to employ an energy *model*. The inputs to the model are counts of the various instructions. To compute the energy from this count data: multiply the count by the EPI for the instruction. Instructions naturally fall into certain categories based on their EPI. Memory-referencing instructions, however, behave differently according to whether the reference was in cache or not. So, a more refined energy model would count the number of cache hits/misses. An even further refinement divides the cache hits/misses into each cache level in the memory hierarchy. A reasonably refined energy modeling tool, then, consists of 2 elements:

1) A method to count the various instruction categories, including all memory-reference refinements.
2) Data on the EPI of the various instruction categories.

The instruction-counting element of the energy-modeling tool uses *Cachegrind*, a *Valgrind* tool [16] as a basis for counting the cache effects of the program under study. Unfortunately, *Cachegrind* is restricted to 2 levels of cache hierarchy. So L2 and L3 cache hits/misses are lumped into the Lm category of cachegrind. Also, Cachegrind does not distinguish vector instructions from scalar instructions. So, for the moment, our instruction counting mechanism uses *Cachegrind* to count *only* the cache behavior of the memory-referencing instructions. The microbenchmarks we used were sufficiently small that we could simply look at the loop body to count non-memory instruction classes by hand. When larger programs are under consideration, we are planning to augment Cachegrind to

distinguish instruction class in the counts.

For the 2nd element of our energy modeling tool, we obtained the EPI of almost all the instruction classes by using the RAPL microbenchmark techniques from section IV. Those results are shown in Table II. The only non-measured results in the table are the L2 and L3 cache costs.

A. Abstracting Energy Costs through Virtual Joules

In order compare energy savings techniques across several processors, Intel core i7 used in the previous measurements, A virtual Joule (vJ) model is developed in to order to estimate the relative energy savings. This model simplifies the energy estimation task by regrouping several instruction having equivalent energy costs, normalized to the least expensive operations. Table II also shows the Virtual Joules, though in this particular case vJ= nJ/ 2. Further details on this method can be found in [4].

B. Analyzing the Fused Multiply-Add Microbenchmark

We used cachegrind to count the number of cache hits/misses for 2 levels of cache (L1 is separate from L2 and L3).

Non-data instructions, i.e. vector arithmetic operations, and vector conversions (half precision to single precision and reverse) could easily be counted directly in the source code of each microbenchmark.

The instruction distribution of each microbenchmark is plotted fig. 2. For all the three microbenchmarks cache hits are mostly L1 hits. The numerical counts of the instruction classes appear in Table III.

The estimated total energy consumed by each benchmark is presented Fig. 3. The modeled energy matches the measured energy very well. That means we can use our model to further analyze the energy effects. Note that the table shows that single precision executes half the instructions that double precision executes. That is a direct effect of vectorization: single-precision vectors are twice as long as double-precision vectors. Therefore, the energy gains are about a factor of 2. There is, however, an additional effect. Notice that the fraction of cache misses is smaller by about a factor of 3. The combination of these 2 effects leads to a factor of 3 in savings.

When comparing single to half, note that the number of memory references is the same. Half precision, however, must execute the conversion instructions. Notice, however, that the cache misses for half-precision are reduced by a little more than a factor of 2. This favorable cache behavior is enough to overcome the conversion instructions plus produce a savings of about 0.20.

We are still investigating if the vectorization effect of half precision can be exploited.

VI. Lessons Learned and Remarks

The important lesson from this study is that the energy gains derived from reduced-precision computation have three sources:

TABLE II: Measured EPI

Instruction	EPI (nJ)	Virtual Joules(vJ)
Integer ADD (scalar)	2	1
Single ADD (scalar)	4.9	2.45
Double ADD (scalar)	5	2.5
128 Vector ADD (singles)	7.5	3.75
256 Vector ADD (singles)	9	4.5
NOP	1.4	0.7
Half to Single vector conversion (8 values)	12	6
Memory Integer load (scalar)	183	91.5
Memory Single load (scalar)	195	97.5
128 vector load	199	98
256 vector load	204	102

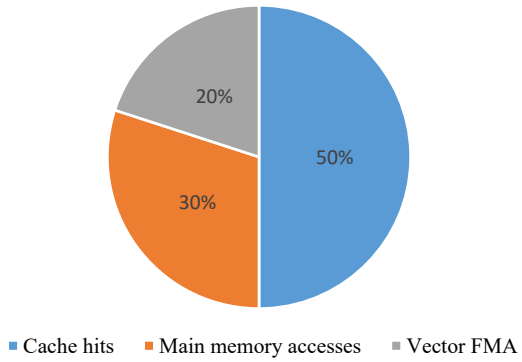
TABLE III: Instruction count for the three microbenchmark running as single processes with an array of 67 million elements and 44 consecutive runs

	Double	Single	Half
Cache hits	1,845,496,308	1,107,298,372	1,291,849,672
Main Memory	1,107,296,473	369,098,968	184,549,898
Vector FMA	738,197,504	369,098,752	369,098,752
Vector convert	0	0	1,476,395,008

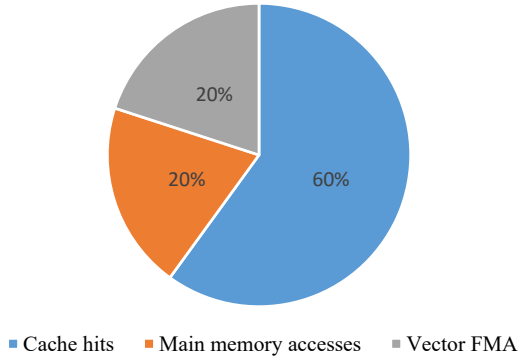
1) Vectorization of arithmetic: Twice as many operations are performed per cycle for single, as compared to double.
2) Vectorization of memory accesses: Smaller precision means more elements per load/store. This is the biggest source of gains.
3) Improved cache utilization: Smaller precision means more values can fit in the cache. This increases the number of cache hits.

The main gain comes from the reduction in memory traffic. As communication is expected to consume an increasing fraction of the total compute energy, the savings will be even more significant in the future. An added advantage of lower precision is the need for less memory, which is an important consideration, since future high-performance computing systems are likely to have a worse flop to DRAM ratio than do current ones.

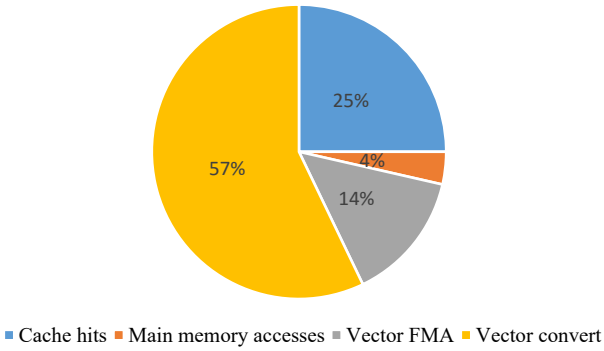
While the potential for savings as shown in this paper are interesting, the opportunity that we see here is an ability to use precision to virtually reduce the effects of technology scaling. Thus, we believe that it will be very interesting to consider algorithms being *redesigned* so that the saved energy can be *reinvested* and used to actually improve the application's quality. Classical supercomputing workloads that have the flavor of iterative solutions [17] as well as weather and climate models [18] are prime candidates for using inexactness to save in the first instance through lowered precision in the first instance, but then reinvest the saved energy in a different part of the algorithm to improve the overall quality. Consequently, the total energy budget will be the same but however the application could be re-engineered to achieve a higher quality solution through this approach. If successful, this approach can be used to mitigate the many hurdles and concomitant costs associated with deploying the next generation of supercomputers by effectively achieving the quality goals that such scaling would imply from an application standpoint!



(a) Double-precision microbenchmark



(b) Single-precision microbenchmark



(c) Half-precision microbenchmark

Fig. 2: Pie chart of the instruction distribution for the three vector Fused Multiply and Add (FMA) microbenchmarks

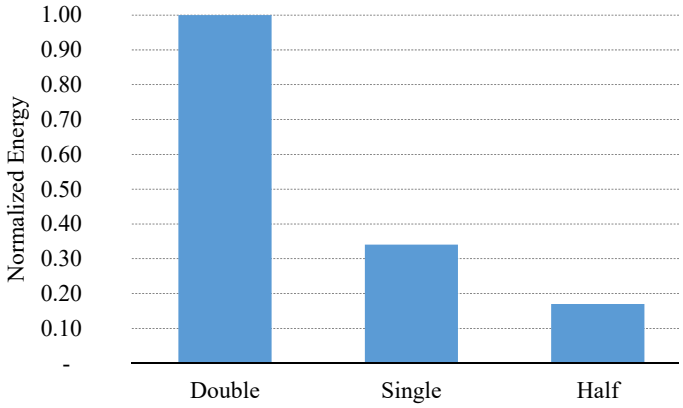


Fig. 3: Normalized energy consumption of the microbenchmarks estimated by the power model for a single-process execution with an array of 67 million elements

REFERENCES

[1] "National strategic computing initiative strategic plan," July 2016. [Online]. Available: https://www.whitehouse.gov/sites/whitehouse.gov/files/images/NSCI Strategic Plan.pdf

[2] K. V. Palem, "Compilers, architectures and synthesis for embedded computing: retrospect and prospect," in *Record of the IEEE W. Wallace McDowell Award Lecture, Proceedings of the ACM-IEEE Inernational Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, 2010.

[3] K. Palem and A. Lingamneni, "Ten years of building broken chips: The physics and engineering of inexact computing," in *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 2s, May 2013, pp. 87:1–87:23.

[4] P. Duben, J. Schlachter, Parishkrati, S. Yenugula, J. Augustine, C. Enz, K. Palem, and T. N. Palmer, "Opportunities for energy efficient computing: A study of inexact general purpose processors for high-performance and big-data applications," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE '15)*, 2015, pp. 764–769.

[5] K. V. Palem, "Proof as experiment:probabilistic algorithms from a thermodynamic perspective," in *Proceedings of the Intl. Symposium on Verification (Theory and Practice)*, June 29-July 4, 2003.

[6] S. Cheemalavagu, P. Korkmaz, and K. V. Palem, "Ultra low-energy computing via probabilistic algorithms and devices: CMOS device primitives and the energy-probability relationship," in *Proceedings of the 2004 Intl. Conference on Solid State Devices and Materials (SSDM), September 14-17, 2004.*, 2004.

[7] K. V. Palem, "Energy aware computing through probabilistic switching: A study of limits," *IEEE Transactions on Computers*, 2005.

[8] K. Palem and A. Lingamneni, "Ten years of building broken chips: The physics and engineering of inexact computing," *ACM Transactions on Embedded Computing Systems*, 2013.

[9] Y. S. Shao and D. Brooks, "Energy characterization and instruction-level energy model of Intel's Xeon Phi processor," in *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*, 2013, pp. 389–394.

[10] C. R. Gonzalez, C. Nguyen, H.-D. Nguyen, J. Demmel, W. Kahan, K. Sen, D. H. Bailey, C. Iancu, and D. Hough, "Precimonious: Tuning assistant for floating-point precision," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13)*, 2013.

[11] A. Buttari, J. Dongarra, J. Kurzak, P. Luszczek, and S. Tomov, "Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy," *ACM Transactions on Mathematical Software (TOMS)*, vol. 34, no. 4, p. 17, 2008.

[12] D. Hackenberg, T. Ilsche, R. Schöne, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: A quantitative comparison," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 194–204.

[13] C. Lomont, "Introduction to intel advanced vector extensions," *Intel White Paper*, 2011.

[14] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[15] P. Kogge and J. Shalf, "Exascale computing trends: Adjusting to the "new normal" for computer architecture," *Computing in Science and Engg.*, vol. 15, no. 6, pp. 16–26, Nov. 2013.

[16] "Valgrind Home." [Online]. Available: http://valgrind.org/

[17] J. E. Dennis, Jr and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[18] T. Palmer, "Modelling: Build imprecise supercomputers," *Nature*, September 29 2015.