LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-795750

Variable Precision Computing

J. A. Hittinger, P. G. Lindstrom, H. Bhatia, P. T. Bremer, D. M. Copeland, K. K. Chand, A. L. Fox, G. S. Lloyd, H. Menon, G. D. Morrison, D. Osei-Kuffuor, N. T. Pinnow, D. J. Quinlan, G. D. Sanders, M. Schordan, T. Vanderbruggen, D. Hoang, P. Klacansky, V. Pascucci, W. Usher, M. Lam, L. G. Moody, J. D. Diffenderfer, A. Metere, L. M. Yang

October 31, 2019

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Variable Precision Computing

Jeffrey A. F. Hittinger (17-SI-004)

IM release # LLNL-TR-795750

Abstract

This report summarizes the activities and major accomplishments of the Variable Precision Computing Strategic Initiative project. The overarching goal of this project was to initiate and promote a new paradigm in High Performance Computing (HPC) that would fundamentally change how we represent and make use of representations of real numbers in finite precision and that would establish Lawrence Livermore National Laboratory (LLNL) as a leader in precision-related research for HPC. We pursued three integrated and concurrent research thrusts: new, more dynamic data representations to address data motion and capacity limitations; improved mixed precision algorithms to accelerate computation and to leverage new hardware; and new tools to help developers reason about precision and to automate code transformations. Within these thrust areas, we made significant advances in the use of compressed array data types in numerical calculations, in the theoretical justification, in improved performance, and in the prospects of hardware implementation; in the development of metrics to significantly reduce the storage needs of molecular dynamics simulations; in the development and analysis of efficient mixed-precision algorithms for time-dependent partial differential equations and for orthogonal factorization; and in the capability of tools that can analyze sensitivity of computed results to choices in precision and that can make automated code transformations based on such analysis. We established that there are definite opportunities to realize significant improvements in HPC above and beyond the 2x limits of traditional mixed precision using the technologies we have developed. The project produced numerous papers, presentations, and posters in important venues as well as open-source software contributions to the community, and many of the new capabilities have initiated or are being leveraged in ongoing projects.

Background and Research Objectives

Within a computer, real numbers are represented using a finite number of 0's and 1's, i.e., finite precision, which means there are a finite number of exactly representable numbers depending on the number of bits used. Decades ago, when memory was a scarce resource, computational scientists routinely worked in 32-bit (single) precision and were more sophisticated dealing with the pitfalls of finite-precision arithmetic. Today, we routinely compute and store results in 64-bit (double) precision by default, even when very few significant digits are required of a calculation. More precision is often used as a simple guard against corruption from finite-precision roundoff error instead of making the effort to ensure algorithms are robust to roundoff. In other applications, only isolated calculations, like tangential intersections, require extended precision.

As indicated in Figure 1, many of the 64 bits represent errors – truncation, iteration, and roundoff – instead of useful information about the solution. This over-allocation of resources wastes power, bandwidth, storage, and Floating-Point Operations Per Second (FLOPS): many meaningless bits are stored, communicated, and computed. Today, we are at a crossroads where increases in compute power based on traditional technologies are dwindling due to physical limits, so to continue to increase performance, we need to reconsider inefficiencies in our approaches. Indeed, many

physical simulation codes obtain only a few percent of peak FLOPS performance because data motion is too slow to keep processors busy with useful work.

Because of the growing disparity of FLOPS to memory bandwidth and capacity (Lucas et al. 2014), the rise of General-Purpose Graphics Processing Unit (GPGPU) computing, and the introduction of special-purpose, high-performance, low-precision hardware like NVIDIA's Tensor Core Unit (NVIDIA 2019), there has been renewed interest in mixed precision computing, where tasks are identified that can be accomplished by using reduced precision (half or single) in conjunction with double precision. Such static optimizations reduce data movement and FLOPS, but their implementations are time consuming and difficult to maintain, particularly across computing platforms. Task-based mixed-precision would be more common if there were tools to simplify development, maintenance, and debugging, but it inherently is limited by not being able to adapt to the precision needs of a calculation.

We often dynamically adapt discretization parameters (mesh size and approximation order) and models in physics simulations to focus the greatest effort only where needed. In the Variable Precision Computing (VPC) project, our ambitious goal was to do the same with precision: to develop technologies that would allow simulations to adjust dynamically precision at a per-bit level depending on the needs of the task at hand. Just as adaptive mesh refinement (AMR) adapts spatial grid resolution to the underlying solution, our system can locally provide more or less precision as needed. An overarching goal was to establish Lawrence Livermore National Laboratory (LLNL) in a leadership role in what we perceived to be an important growing direction in High Performance Computing (HPC) research.

Acceptance from the community necessitates that we address three concerns: that we can ensure accuracy, ensure efficiency, and ensure ease of use in development, in debugging, and in application. To achieve our vision, we pursued three integrated and concurrent thrusts that pulled together recent advances from multiple areas: new, efficient data representations to address the data motion and storage problems; improved mixed precision algorithms to reduce unnecessary computation and to leverage specialized hardware; and new tools to help developers reason about precision and automate data type transformations. Underlying these thrusts was a significant effort in numerical analysis to provide the necessary theoretical justification for our work. Our expectation was that VPC would allow for 4-100x less data storage and would increase computational throughput by factors of 2-10x for bandwidth-limited applications.

Specifically, our original intent was to consider a breadth of high-level research objectives. Building on the ZFP compressed floating point array representation developed at LLNL, we sought to couple this format with a hierarchical multiresolution data format to provide a "data optimal" representation that could refine or coarsen mesh resolution and precision as needed to minimize data under an error budget for visualization and data analysis purposes. We also planned to develop a rigorous analysis of the roundoff error accumulation for calculations in which the underlying data was represented in the ZFP compressed array format and to extend the ZFP representation to be locally adaptive to further increase the effective gains in data reduction. To reduce computation, we sought to extend the error transport approach for a posteriori error estimation from truncation error to roundoff error estimation and to consider combining it with a local, patch-based approach related to AMR. Furthermore, we sought to investigate the utility of reduced (mixed) precision algorithms for eigensolver problems frequent in molecular dynamics and graph clustering and to demonstrate these algorithms on a new ARM/GPU cluster to be procured by Livermore Computing (LC). On

the tools front, our major research objective was to use compiler-based, source-to-source transformation techniques to develop tools to automatically convert types consistently in codes to ease the burden of software maintainability of switching types or developing mixed precision implementations. Throughout the project, we sought to demonstrate our techniques on LLNL-relevant application codes, with a splash app of demonstrating the adaptive ZFP capability in the radiation transport code Ardra.

We successfully completed the majority of these objectives, even as the scope and direction of the research naturally evolved. Notably, we did not complete a demonstration of mixed precision algorithms on an ARM/GPU cluster nor did we fully complete a demonstration of ZFP compressed arrays on the full Ardra application problem, both of which were end goals of lengthy research paths. In both cases, aligning tasks to expertise within the team proved to be a limiting factor, in addition to the unexpected challenges that arise in any research project. However, in addition to the original objectives, we pursued promising new research directions as they arose. We produced additional analyses of (and corrections for) the bias in ZFP lossy compression, demonstrated linear equivalence of error transport with iterative refinement and identified additional opportunities for parallelization, and extended traditional roundoff error analysis methods to account for mixed precision. In our pursuit of efficient means of reducing data from unstructured particle methods, we investigated a number of reordering techniques in ZFP as well as the concept of channel capacity as a metric for decimation of molecular dynamics simulation data. Our work on tools also grew to accommodate algorithmic differentiation techniques to identify candidate variables for lower precision, and ultimately, we developed a more sophisticated analysis tool chain than originally planned.

Scientific Approach and Accomplishments

Within each of the three thrusts of our project (new data representations, improved mixed precision algorithms, and enabling tools), there were multiple research threads. We here summarize the approaches and accomplishments of each major effort within the thrusts.

New Data Representations

One thrust of our project dealt with new, more efficient ways of representing data. For data analysis, we investigated data optimal representations that could locally adapt both the precision of the data and/or the granularity of the underlying grid based on the needs of the analysis. We investigated the ZFP compressed data array theoretically, bounding the lossy-compression errors and approximation bias; in hardware, developing a logic gate layout suitable for FPGA use; and in practice in a LLNL-relevant transport algorithm. We also developed an adaptive form of ZFP, Adaptive Rate Compression (ARC), which is even more efficient in its ability to adjust storage needs for specified levels of accuracy. We formulated a universal coding of the reals, a superset of real number representations including the IEEE (Institute of Electrical and Electronics Engineers) 754 floating-point standard and implemented these generic types in software so that they can be evaluated in application codes. Finally, for problems with little data coherence like particle methods, we used ideas from information theory to develop metrics that can be used to accelerate calculations and significantly reduce data storage needs.

Data Optimal Representations for Analysis

Traditionally, there are two, somewhat orthogonal, strategies to reduce data sizes: reducing the resolution (subsampling or adaptive representations) or reducing the precision (quantization or

compression). Working with colleagues at the University of Utah, we combined the two approaches to create a new paradigm of data reduction, where application-specific heuristics can be used to choose between discarding the least-important data value (i.e., reduce resolution) or discarding the least-important bits from a set of less-important values (i.e., reduce precision) (Hoang et al. 2018). We developed a new resolution-precision-adaptive, in-memory data representation that can ingest these arbitrary bit orderings to provide an interface to existing tools and algorithms: Mixed-Precision Adaptive Multilinear Meshes (MAMMs), which use multilinear B-spline wavelets to create a spatially adaptive mesh of rectangular cells and which represents function values on this mesh using spatially coherent blocks of reduced precision in increments of bytes (Bhatia et al. 2019). Combined, these two features enable resolution-precision adaptivity for scientific data.

MAMMs is the first data structure to support ingesting arbitrary data streams, and it provides superior spatial adaptivity and ensures C^0 continuity without explicit representation of T-junctions. We demonstrated that resolution-precision streams perform better than resolution-only and precision-only streams and that streams tailored to the specific applications outperform generic resolution-precision streams. The improved adaptivity using MAMMs can provide more than a 2x reduction in the memory footprint of scientific data and can be easily interfaced with existing tools, such as VisIt.

ZFP as a Compressed Data Type

An obvious approach to address the challenges of data motion and capacity is to consider data compression techniques that reduce the amount of data and, thus, time in data transfer. Lossless compression algorithms struggle to produce significant compression ratios for floating-point data, but lossy floating-point compression produces a much higher ratio of data reduction. However, as lossy compression techniques fundamentally introduce error, in order to be widely adopted, theoretical justification is needed to ensure that the reconstruction of the compressed data retain sufficient accuracy for the intended purposes. While prior investigations into the use of lossy compression techniques for scientific were limited to input and output, a critical departure in our work was to promote the use of ZFP compressed arrays as an alternative data type to be used to store simulation data in situ, that is, to store the data in a lossy compressed state, convert (decompress) small blocks of the data into double precision, operate on the data, and then apply lossy compression to store the results. The solution state already contains truncation, iteration, and other roundoff errors, so as long as error from these compressed types is bounded and below other errors, there is no change in the meaningful digits of accuracy.

We were able to provide the first closed form expression of the error caused by ZFP by considering a generalization in the space of infinite dimensional bit vectors – a novel approach to such an error analysis (Diffenderfer et al. 2019). Our methodology can be generalized to any other algorithm involving the manipulation of bits, which could have a vast impact on future of floating-point round-off analysis. We used these techniques to identify the sufficient requirements of advancement operators to ensure that error accumulation caused by the repeated application of ZFP is provably bounded and also developed forward and backward error analysis to provide requirements on the fixed precision parameter, an input into the ZFP algorithm (Fox et al. 2019). A concern that arose for simulations of chaotic phenomena, e.g., climate, was that the ZFP compressed data type did not introduce a systematic bias through repeated application. We proved rigorously that ZFP did indeed possess a bias and were able to modify the compression algorithm to significantly reduce the bias.

ZFP in Hardware

The promise of ZFP compressed data types is currently limited by their implementation in software; performance gains are limited to the spare compute cycles available before reaching the bandwidth limitations of the communication channel. A hardware implementation of ZFP has the potential to raise the bar on performance, and from the inception of ZFP, it was designed to accommodate a hardware implementation (Lindstrom 2014). Ideally, ZFP could be supported in hardware at the processor cache so that data are moved and stored compressed and only decompressed for processing.

As a proof of concept, we developed a hardware implementation of the ZFP compression algorithm in SystemC to facilitate its evaluation in various architectures. In this implementation, which supports 1D, 2D and 3D blocks of floating-point numbers, the best performance is realized from the hardware ZFP unit when batches of blocks are processed at a time; see Figure 2. The SystemC ZFP implementation can target an FPGA or be added to a new chip design. Latency for the current implementation can be as low as 34 cycles in the 1D case and up to 172 cycles in the 3D case, which translates to a speedup of the hardware implementation over the software from about 15x (1D) case to over 200x (3D). With these speed-ups, the additional computational cost of ZFP (de)compression in hardware would be an insignificant overhead, unleashing a larger percentage of the true processing power of modern architectures.

Adaptive Rate Compression (ARC)

We have shown that, depending on the application, numerical data can be reduced by 4-100x with acceptable error using the ZFP "lossy" compression. One drawback of ZFP compressed arrays is that they allocate the same number of bits uniformly over the computational domain, both in space and in time, resulting in an excess in accuracy in easy-to-compression regions and insufficient accuracy in regions that do not compress as well but where accuracy is often most needed. Our goal was to represent the whole domain at uniform accuracy (or precision) while still taking advantage of compression. By spatially and temporally adapting the compression ratio to meet a prescribed error tolerance, bits could be allocated more intelligently, ultimately ensuring higher accuracy in quantities of interest for the same storage budget.

We developed a new ZFP-based data structure, called ARC (Adaptive-Rate Compression), that provides orders-of-magnitude higher accuracy than either ZFP or IEEE standard types in numerical computations for the same storage or, alternatively, less storage for the same accuracy. Our ARC data structure has been carefully designed to store only the minimum amount of information needed to locate variable-sized compressed blocks. The outcome of this effort is a new variable-rate compressed-array data structure that transparently handles memory management associated with block allocation, bookkeeping, and (de)compression and caching of blocks. We have developed a complete implementation of adaptive-rate compressed arrays that supports ZFP's fixed-precision mode (controlling local relative error); its fixed-accuracy mode (bounding absolute error); and its reversible mode (lossless compression). ARC has been evaluated in a volumetric raytracing application as well as in the Euler2D shock physics code. As shown in Figure 3, we demonstrated orders-of-magnitude reductions in error for a given storage size, which, alternatively, can translate into effective compression rates larger than 4x.

ZFP Demonstration in a Transport Code

ZFP and ARC are currently being evaluated within Kripke, a mini-App designed to emulate the discrete ordinate discretization in the Ardra transport code that serves as a model for the core algorithms and data structures inherent to deterministic transport models. The splash application for the ZFP compressed data structure work was originally envisioned to be a specific Ardra calculation. While we did not reach this goal, we did make significant progress in Kripke as a step to implementation in the more complex Ardra code.

For the Kripke effort, our primary goal was to examine the impact of compression on accuracy of the solution with a secondary goal of evaluating the effort required to integrate ZFP in Kripke as a surrogate for a modern HPC code. Our approach involved replacing Kripke's primary data arrays with LLNL's ZFP compressed floating point arrays, which was accomplished by altering Kripke's Field structures as well as some of the underlying RAJA infrastructure. Our implementation was then tested against Kripke's uncompressed results. A variety of parametric studies investigating the compression rate, ZFP cache size, and test problem type allowed us to extract the performance data needed to assess the impact of fixed-rate compression on Kripke.

For problems without scattering, essentially a test of a single "sweep" in the algorithm, the underlying discretization errors dominate the overall error at compression rates above 4 bits/value. For example, on a problem with 2.88 billion zones and 32 directions, a compression rate of 8 bits/value used only 16% of the memory required by the uncompressed code yet had error norms matching the uncompressed model to several significant digits. For problems with scattering, the results are the same: compression rates of 4 bits/value and above match the solution norms of the uncompressed problem to many significant figures.

Universal Coding of the Reals

While IEEE floating point has served us well for decades, theoretical analysis and experimental evidence point to several drawbacks. Novel number representations beyond IEEE, like Gustafson's posits (Gustafson 2017), promise a better tradeoff between precision and dynamic range that may allow many computations to be performed at lower precision, reducing memory footprint, data movement, and power usage. We sought to investigate new number representations that are simpler to encode and decode than IEEE, that are better tuned to the empirical distribution of values that occur in HPC codes, that support variable precision by gradually extending both the dynamic range and precision at single-bit granularity, and that allow encoding of any real number given sufficient precision.

To empirically understand the trade-offs of numerous choices in LLNL-relevant applications, we developed two independent frameworks. Our NumRep framework supports fast, variable-length encoding of exponents, which allows trading exponent and mantissa bits to both maintain dynamic range and to expand precision of common numbers. NumRep also supports user-defined fraction maps that translate mantissa bits to real fractions and rules for overflow, underflow, and rounding. Our other framework, Flex, supports more general number systems by expressing numbers via binary subdivision rules of intervals (Lindstrom 2019), allowing number systems – including all systems investigated on this project – to be defined in only two lines of code. Our Flex library is more powerful than NumRep, but also less performant. Since our experimental number types are not supported in hardware or natively by the compiler, a C++ wrapper library for floating-point

operations was developed using template metaprogramming to supply new floating-point types and to perform operations on those types. In addition, the templates have been instrumented to capture operations and gather statistics on floating-point applications compiled with the library.

Our frameworks unify several known but seemingly unrelated representations under a single schema, including IEEE floating point and posits (Lindstrom et al. 2018). The NumRep framework, in particular, allows for independent experimentation with exponent codes, fraction mappings, reciprocal closure, rounding modes, handling of under- and overflow, and underlying precision. One discovery was that, by generalizing the Elias universal codes for positive integers (Elias 1975) to the whole real number line, the Elias gamma code coincides with one member of the posit family. Through application to various linear algebra and shock hydrodynamics application (Figure 4), we demonstrated that number representations with variable-length exponents consistently outperform those with fixed-length-exponent representations. Representations like posits and Elias codes generally have greater arithmetic closure, smaller mean representation error, more efficient representation of infinity and NaNs (Not a Number) and produce lower roundoff error – often by at least an order of magnitude – than IEEE types.

Identifying Information in Data

Most compression techniques rely on underlying coherence in data to reduce the amount of data stored. In some sense, coherence (smoothness) serves as a measure of the local information content of the data. Some data, such as particle information, generally lacks coherence – at least in low-dimensional spaces – and other measures are needed to quantify the local information content. Being able to compress or reduce particle simulation data to only that of highest information content would be revolutionary: Molecular dynamics (MD) simulations, e.g., currently can produce several Terabytes to Exabytes of data per run. However, in many of these data files, there is little of interest; nothing of significance is contained therein until a rare event occurs. If the occurrence of a rare event could be identified and then only output the most important data from an MD simulation, orders of magnitude of data savings would occur.

Using ideas from Information Theory, we developed techniques that can enable much better data handling in MD simulations. The on-the-fly detection of the rare events has been addressed by creating new mathematical models capable of measuring the Shannon-Hartley channel capacity of a classical Hamiltonian system of the kind simulated with classical MD. This measure also allows us to partition the simulated system in interesting regions to be stored and uninteresting regions to be discarded (classification) and to define classes of relevance based on the amount of precision needed to save the state of each particle. We have also investigated techniques to adjust time stepping in such a way that more time resolution is used in interesting regions, while uninteresting regions are calculated less often, resulting in performance increases. The detection capabilities have been implemented as part of the input script for the MD simulation software, LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator), which is widely used at LLNL, and the time step adjustment based on channel capacity has been implemented in the position and velocity update routine of LAMMPS/Kokkos and is currently work in progress.

Improved mixed precision algorithms

While addressing the data transfer and storage problems was the major emphasis of our research, we also investigated methods that could increase efficiency in arithmetic computations by using a combination of low and high precision variables. Specifically, we investigated error transport as a

bootstrapping technique to obtain higher precision from a sequence of low precision calculations. In addition, because of the potential speed-ups using low-precision orthogonal factorization methods in data analysis techniques, we investigated low- and mixed-precision variants of the ubiquitous QR factorization algorithm.

Error Transport

Because of specialized low-precision hardware appearing in HPC systems, there is increasing interest in using low-precision calculations as part of a larger mixed-precision algorithm. The canonical example in this area is the classical defect correction algorithm of iterative refinement, where corrections are computed in low precision and iteratively added to a double precision accumulator until double-precision accuracy is obtained. We were interested in an alternative defect correction method, error transport, which has proven to be useful for a posteriori truncation error estimation (Zhang et al. 2000, Qin and Shih 2003, Banks et al. 2012). We were originally motivated by the idea of using local overlays of data array, in a scheme similar to block structured adaptive mesh refinement, to build up sufficient accuracy in hyperbolic problems. What we ultimately discovered is that the error transport approach has a significant potential as an unrolled – and thus parallelizable – version of iterative refinement.

Error transport is a form of defect correction in which the defect is used to drive an evolution equation for the error. The accumulated error field may then be used to correct the solution at the end of the simulation. We were able to prove that, in the linear case, error transport is equivalent to iterative refinement, where each error correction corresponds to an iteration in iterative refinement. Thus, error transport provides a means to extend iterative refinement to explicit, time-dependent problems, to extend iterative refinement to a nonlinear formulation, and, most importantly, to expose a layer of concurrency that could further accelerate iterative refinement. Since each error correction update can be lagged in time, error transport allows the iterations of iterative refinement to be pipelined and solved concurrently, providing additional speed-up without loss of accuracy. We demonstrated the roundoff error transport strategy on the linear heat equation and the nonlinear porous medium equation. Our preliminary results indicate a 40% speedup over the native double-precision solver using a system of two transport equations discretized in single precision and solved in parallel.

Low-precision Orthogonal Factorization

Orthogonal factorization of a set of vectors is a basic linear algebraic computational kernel used throughout science and data analysis applications. It is used as a subroutine for many linear algebraic solver algorithms (e.g., iterative singular value and eigensolvers) or for analyzing statistical correlations in data (e.g., principal component analysis). However, in many data analysis applications, the accuracy of the orthogonality is not required to perform the analysis task with high accuracy; low-precision arithmetic can produce useful results. For example, we have demonstrated the use of low precision for a class of spectral graph clustering applications that use approximate eigenvectors to embed graph vertices into a low-dimensional space.

We developed and analyzed low- and mixed-precision QR factorization routines, which take a matrix of column vectors A and compute a nearly orthogonal matrix Q and upper-triangular matrix R such that $A \approx Q \bullet R$. We developed a custom, low-precision Modified-Gram-Schmidt QR Algorithm and modified a Tall and Skinny Householder QR (TSQR) algorithm from the literature to accommodate low and mixed precision; testing and analysis narrowed our interest to the more

robust, pre-existing technique TSQR. A significant contribution of our work was to generalize the analysis techniques from (Higham 2002) to accommodate mixed precision. With these new techniques, we performed in-depth analysis of mixed precision TSQR for general floating-point formats (Yang et al. 2019). Several experiments on manufactured problems with controllable condition numbers and on spectral clustering and dynamic systems revealed that TSQR often can increase accuracy in low-precision arithmetic over Householder QR (HQR), or other non-blocked methods, due to using lower dimensional inner-products. We derived worst-case bounds for accuracy of mixed-precision HQR and TSQR and used these to characterize when TSQR should be used over the traditional HQR. Finally, we demonstrated that low and mixed precision is often successful in two data analysis applications: spectral graph clustering and discovering governing equations in dynamic systems.

New tools to reason about and automate precision transformations

Refactoring codes to include new data representations or to make use of mixed-precision algorithms entails a level of effort that serves as a barrier for adoption. To aid developers, we have developed tools that can provide helpful information about algorithmic sensitivity to precision choices (ADAPT) and that can automate the process of type conversion while ensuring consistency with dependencies (Typeforge). In combination with one of our academic collaborators, we have developed a tool chain (FloatSmith) that can perform an automated precision sensitivity analysis on source code and provide transformed, mixed-precision code that provides improved performance.

Typeforge

Modern computer architectures support multiple levels of precision when performing floating point arithmetic operations, and different choices have a wide impact on performance as operation latency, memory footprint, and communications increase. Thus, there is increasing interest in developing mixed-precision applications from the existing high-precision versions. In this case, applications are refactored to use lower precision when it does not harm the accuracy.

Reducing the precision in large application is time consuming with complex behaviors and many possible changes. Typeforge automates the process of changing types at the source-level, performing any of these changes on the source code without human intervention. Typeforge is a source-to-source tool based on the ROSE Compiler (Quinlan and Liao 2011) that detects any potential changes and analyzes how these changes interact with each other. For example, when one changes the precision of an array, one must also change the code use to allocate that array. We developed a new algorithm to determine the set of all changes necessary when the type of one C/C++ language construct is changed. Our algorithm automatically computes clusters of such changes. The Typeforge clustering ability became essential when we transitioned from HPC benchmarks to the LLNL proxy-app LULESH. A type change on a single variable can cause a cascade of changes to other variables' types to obtain a correct program. In LULESH, Typeforge found 755 possible changes, but consolidated the search space to 394 consistent clusters of changes. The large number of changes that may be required to achieve mixed-precision shows why it is not practical to refactor manually.

ADAPT

Developers must take care to ensure that the roundoff errors introduced in mixed precision implementations are within the acceptable thresholds so as not to corrupt the output. Large errors can render results useless. Unfortunately, it is not always easy to develop mixed-precision versions

of large codes manually, as this requires both knowledge of the numerical behavior of the algorithm and also an understanding of the subtle details of floating-point rounding errors.

We proposed an approach, implemented in a tool called ADAPT (Algorithmic Differentiation Applied to Precision Tuning), that uses algorithmic differentiation, a method for numerically computing the derivative of computer programs, to analyze floating-point precision sensitivity of outputs to variables and operations in a program. The results can then be used to inform the development of a mixed-precision version of the program. Our technique enables the scaling of rigorous precision analysis techniques to benchmarks and proxy applications, which is an important step toward the application of mixed-precision analysis to full-scale HPC applications.

We compared ADAPT to state-of-the-art approaches on several benchmarks and mini-applications; our evaluation showed that ADAPT is able to estimate the output error of mixed-precision programs accurately and significantly faster than other existing tools. ADAPT has a key capability to estimate the output error for every dynamic instance of the variable, which allows it to find temporal and spatial regions that can be represented in lower precision. This insight was used in HPCCG (HPC Conjugate Gradient), a Mantevo benchmark, where the code was modified to use higher precision in the initial iterates and then switch to lower precision arithmetic in the later iterations to achieve 1.1x speedup without losing any accuracy. Detailed analysis provided by ADAPT was used to transform LULESH, which resulted in 20% speedup without loss of accuracy; ADAPT performed significantly better compared to other state-of-the-art mixed-precision tools that can be evaluated on LULESH (Menon et al. 2018).

FloatSmith

The goal of the FloatSmith toolchain was to provide an open-source, automated, end-to-end source-level mixed-precision tuner that leverages that capabilities of several software tools into an integrated tool chain. The toolchain consists of CRAFT (Configurable Runtime Analysis for Floating-point Tuning), developed at James Madison University; Typeforge; and ADAPT. CRAFT is general framework for floating-point program analysis that supports several search strategies including combinational, compositional, and delta-debugging. We leveraged the existing testing-based search framework of CRAFT to implement source-level tuning.

Figure 5 shows an overview of the integrated tool chain. We use Typeforge to extract a list of tunable variables (along with clusters or groups based on static type dependency analysis) and, optionally, to insert ADAPT instrumentation. If ADAPT is used, the user must also insert some pragma-based annotations to describe the program outputs and allowed error thresholds. The ADAPT-instrumented program runs and produces a mixed-precision recommendation that is guaranteed not to exceed the user-provided error threshold; however, it might not speed up the program. Finally, CRAFT performs a variable configuration space search (optionally starting from the ADAPT results or the Typeforge variable clusters/groups) and attempts to find a mixed-precision configuration that both passes a user-defined representative workload and verification routine and achieves a speedup. If such a configuration is found, CRAFT reports the fastest and provides the modified source code. Each configuration is built using Typeforge, and the search process can parallelize naturally if multiple cores or nodes are available.

FloatSmith was applied to several benchmarks. The toolchain was able automatically identify a mixed-precision configuration that gives 80% speedup for the vector-vector add-multiply

benchmark AXPY. FloatSmith was also able to fully automate mixed-precision tuning of the LLNL proxy app LULESH. However, the automatic tool chain gave only a 2% speedup for a one-digit error threshold (cf. 20% speedup for 9-digits with ADAPT). We plan to improve FloatSmith to support more sophisticated transformations that would enable larger speedups.

Impact on Mission

HPC permeates how the Laboratory meets its missions, so technologies that fundamentally change the way we compute can have wide-ranging impacts. In this way, the VPC Project has directly advanced the core competencies in High-Performance Computing, Simulation, and Data Science which will benefit of all five mission focus areas and of many of the Laboratory's other core competencies. Because of the ground-breaking work in new data representations and the mathematical theory to support their use, capacity- and bandwidth-limited physical simulations, which include many of the mesh-based and low-order codes at the laboratory, are well-situated to benefit. For calculations not in this regime, the abilities to easily maintain and understand error propagation in mixed-precision code and incorporate mixed-rate compression into data analysis, I/O, and tabular data will still provide significant benefits. More importantly, as demonstrated by the numerous publications and invited talks, the variable precision capabilities developed in this project have firmly established LLNL as the leader in this new paradigm of computing right at a time when the HPC community and vendors are taking a serious look at how we represent and use data (e.g., Intel's new bfloat16 type). In fact, two members of the VPC team were invited to organize a workshop in May 2020 on variable precision computing at the Institute for Computational and Experimental Research in Mathematics (ICERM).

The VPC project has also enriched the Laboratory's research environment. While maintaining a strong collaboration with the University of Utah, we forged a new alliance with Prof. Michael Lam's research group at James Madison University. Over the course of the project, we hired two new postdoctoral researchers and partially supported two others; three of these have already become full staff members in the Center for Applied Scientific Computing, and the expectation is that the third will as well. The project also mentored a dozen summer students, some for multiple internships; one of these has since been hired as a postdoctoral researcher, and a postdoctoral position is being offered to a second.

Conclusion

The VPC Project was very successful in demonstrating the opportunities for rethinking how we represent and compute with finite precision, but, as with all good research projects, there is still more to do. Technically, there are more improvements that can be made to MAMMs, ZFP, and ARC as well as to our theoretical analyses. Our capabilities and initial investigation into the universal encoding of reals have only scratched the surface of understanding the potential trade-offs of different encodings. The power of using channel capacity metrics to help guide precision adaptivity needs further investigation. Typeforge can be further generalized to handle more types and more complex transformations, and ADAPT needs to be generalized to analyze NVIDIA's CUDA language and optimized to reduce memory overhead.

Various aspects of the VPC project will continue on under different funding. We have acquired a small grant for the Department of Energy (DOE) Advance Scientific Computing Research Applied Mathematics Research Program to continue the theoretical analyses of compressed data types. The DOE's Exascale Computing Project and the National Nuclear Security Administration's (NNSA)

Advanced Simulation and Computing (ASC) Program are providing further development and application of ZFP and ARC; nevertheless, we are in negotiations to acquire additional funding from ASC to support completion of the Kripke and Ardra demonstrations. Similarly, we are discussing plans with the institutional common software infrastructure project, RADIUSS, to incorporate ZFP into the CHAI/Umpire memory management abstraction, which would make ZFP more broadly available within the Laboratory. Furthermore, we are discussing with the VisIt team to understand how to best integrate MAMMs to their framework and make it accessible to the laboratory.

ZFP in hardware is a work in progress. Only the encoding or compression half of ZFP is implemented, and decompression is future work. We intend to evaluate ZFP using LiME, our FPGA emulation platform, by translating SystemC into Verilog or VHDL and integrating it with our existing design. We have had multiple discussions with vendors about including ZFP in hardware, and most recently there was intense interest in providing an FPGA design that would allow a vendor to evaluate the technology in conjunction with their own FPGA models.

The tools developed in the VPC Project will continue to be supported and used in future work. Typeforge provides a capability that is essential for LLNL's code modernization effort as it supports old and modern C++, including the RAJA loop abstraction interface, and will be available in the ROSE Toolkit. Typeforge's internal representation of type opens new opportunities for semantic code analysis. ADAPT will be extended for other kinds of approximate computing techniques; a new LDRD-ER was funded to pursue approximate computing in HPC, and ADAPT will be used to identify regions of code where approximate computing techniques can be applied to HPC applications.

The software artifacts of the VPC project are being provided to the HPC community for their use and further development as open source software. ZFP, MAMMs, Typeforge, and ADAPT have all been released, and ARC, NumRep, and Flex are in the process of being released. Our goal in making this software available is to encourage others to begin to investigate precision issues so that we can build a larger, more active community and, ultimately, more momentum for the broader support for and adoption of variable precision computing techniques.

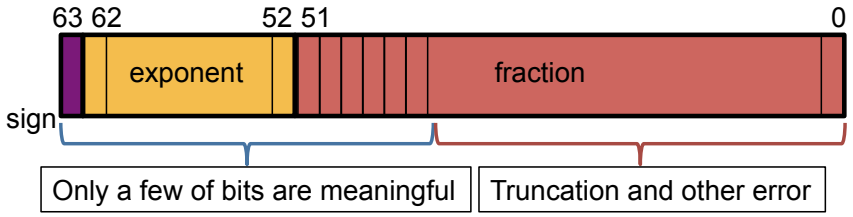

Figure 1: Standard 64-bit floating point binary representation. Because of approximation and modeling errors, only the first few bits of the fraction (mantissa) typically contain relevant information for many calculations. The exponent represents over 600 orders of magnitude; less than 100 orders of magnitude are needed to represent the ratios of largest to smallest scales in the universe.

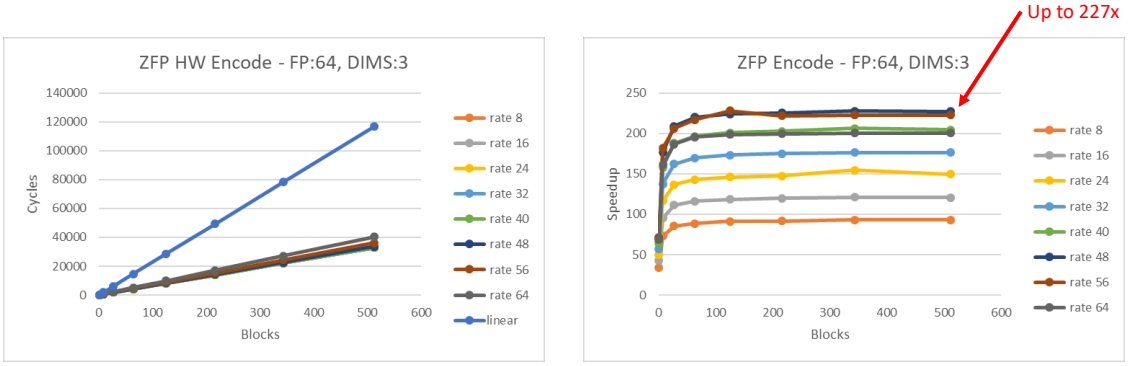

Figure 2: ZFP emulated hardware implementation performance for 64-bit, 3D blocks. (Left) Scaling of cycles with the number of blocks, where "linear" extrapolates the cycle time for one block to many blocks, showing the benefit of encoding batches of blocks through the hardware pipeline. (Right) Cycle-accurate estimated hardware speedup at different rates compared to a single x86_64 core, Intel i7.

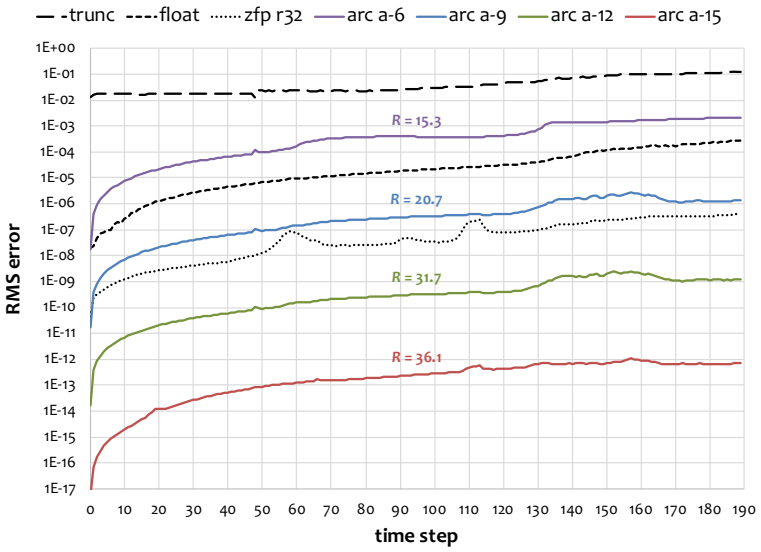

Figure 3: Euler2D application. The truncation error ('trunc') associated with the spatial discretization is on the order of 10^{-2} to 10^{-1}, which is much larger than the roundoff error. We compare adaptive rate compression (ARC) errors in fixed-accuracy mode using absolute error tolerances of $\{10^{-6}, 10^{-9}, 10^{-12}, 10^{-15}\}$ with ZFP in fixed-rate mode and standard 32-bit single precision. The ARC rates are averages over time and include all used storage. At $R = 31.7$ bits/value, ARC achieves 2–5 orders of magnitude smaller errors than ZFP and single precision at $R = 32$ bits/value storage.

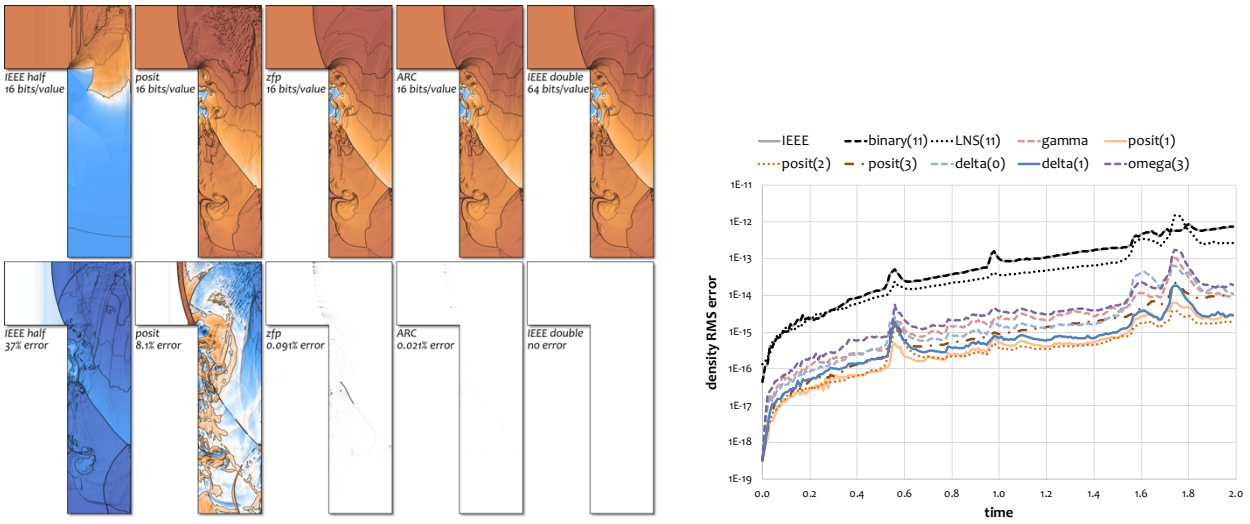

Figure 4: Shock diffraction in an L-shaped channel. (Left) Plots of density solution (top) and error relative to standard 64-bit double precision (bottom) are shown for three 16-bits/value representations with the reference solution. For a quarter of the storage, use of ZFP and ARC (fixed accuracy) compressed arrays produced less than 0.09% and 0.02% root mean square (RMS) relative error, respectively. (Right) RMS error in the density field over time for ten different 64-bit number types computed using NumRep type emulation.

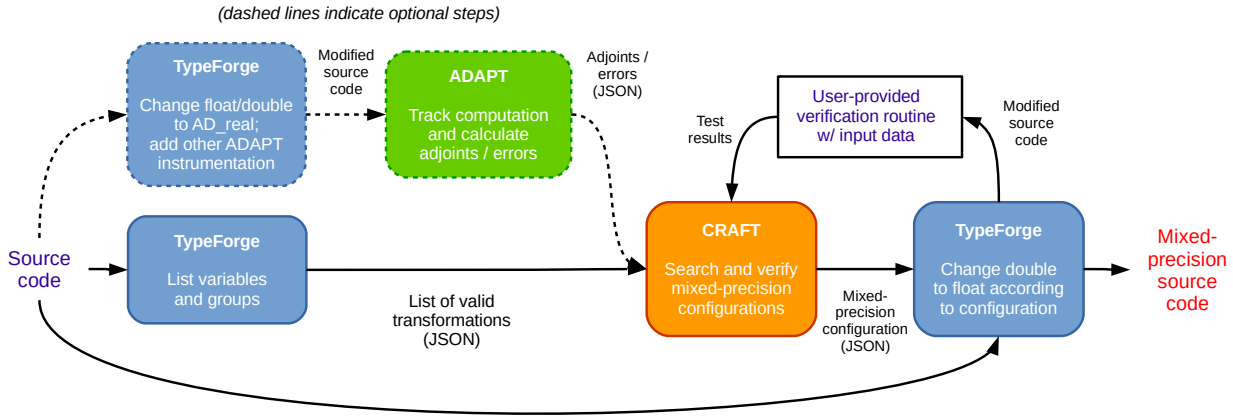

Figure 5: Integrated FloatSmith system tool chain to automate mixed-precision performance tuning and source transformation. The diagram shows the possible interactions between the Typeforge, a source-to-source analysis and transformation tool; CRAFT, a framework for search-baed floating-point program analysis; and ADAPT, a tool that uses algorithmic differentiation to analyze floating-point precision sensitivity of outputs to variables and operations in a program.

References

Banks, J., J. Hittinger, J. Connors, and C. Woodward. 2012. "Numerical error estimation for nonlinear hyperbolic PDEs via nonlinear error transport." *Computer Methods in Applied Mechanics and Engineering* 213-216:891–921. doi:10.1016/j.cma.2011.11.021.

Bhatia, H., D. Hoang, G. Morrison, V. Pascucci, P.-T. Bremer, and P. Lindstrom. 2019. "Mixed-Precision Adaptive Multilinear Meshes." *IEEE Transactions on Visualization and Computer Graphics*. In preparation; planned submission January 2020.

Diffenderfer, J., A. Fox, J. Hittinger, G. Sanders, and P. Lindstrom. 2019. "Error Analysis of ZFP Compression for Floating-point Data," *SIAM Journal on Scientific Computing* 41(3):A1867-A1898. LLNL-JRNL-744818. doi:10.1137/18M1168832.

Elias, P. 1975. "Universal codeword sets and representations of the integers." *IEEE Transactions on Information Theory* 21(2):194–203. doi:10.1109/TIT.1975.1055349.

Fox, A., J. Diffenderfer, J. Hittinger, G. Sanders, and P. Lindstrom. 2019. "Stability Analysis of Inline ZFP Compression for Floating-point Data in Iterative Methods." *SIAM Journal on Scientific Computing*. Accepted with minor revisions, October 2019. LLNL-JRNL-769679-DRAFT.

Gustafson, J. and I. Yonemoto. 2017. "Beating Floating Point at its Own Game: Posit Arithmetic." *Supercomputing Frontiers and Innovations* 4(2):71–86. doi:10.14529/jsfi170206.

Higham, N. 2002. *Accuracy and Stability of Numerical Algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Hoang, D., P. Klacansky, H. Bhatia, P.-T. Bremer, P. Lindstrom, and V. Pascucci. 2018. "A Study of the Trade-off Between Reducing Precision and Reducing Resolution for Data Analysis and Visualization." *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1193-1203. LLNL-CONF-755005. doi:10.1109/TVCG.2018.2864853.

Lindstrom, P. 2014. "Fixed-Rate Compressed Floating-Point Arrays." *IEEE Transactions on Visualization and Computer Graphics* 20(12):2674-2683. doi:10.1109/TVCG.2014.2346458.

Lindstrom, P., S. Lloyd, and J. Hittinger. 2018. "Universal Coding of the Reals: Alternatives to IEEE Floating Point," In Proceedings of the Conference for Next Generation Arithmetic 2018, Singapore, 28 March 2018. In cooperation with ACM. LLNL-CONF-743265. doi:10.1145/3190339.3190344.

Lindstrom, P. 2019. "Universal Coding of the Reals using Bisection." In Proceedings of the Conference for Next Generation Arithmetic 2019, Singapore, 13 March 2019. In cooperation with ACM. LLNL-CONF-764361. doi:10.1145/3316279.3316286.

Lucas, R. et al. 2014. "Top ten exascale research challenges." DOE ASCAC Subcommittee Report. doi:10.2172/1222713.

Menon, H., M. Lam, D. Osei-Kuffuor, M. Schordan, S. Lloyd, K. Mohror, and J. Hittinger. 2018. "ADAPT: Algorithmic differentiation applied to floating-point precision tuning." In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, 11-16 November 2018, Paper 48:1–13. LLNL-CONF-748742. doi:10.1109/SC.2018.00051.

NVIDIA Corporation. 2019. "NVIDIA Tensor Cores." https://www.nvidia.com/en-us/data-center/tensorcore/.

Qin, Y. and T. I.-P. Shih. 2003. "A Method for Estimating Grid-Induced Errors in Finite-Difference and Finite-Volume Methods." AIAA Paper 2003-0845. doi:10.2514/6.2003-845.

Quinlan, D. and C. Liao. 2011. "The ROSE source-to-source compiler infrastructure." Cetus Users and Compiler Infrastructure Workshop, in conjunction with PACT. Galveston Island, Texas, 10 October 10, 2011.

Yang, L., A. Fox, and G. Sanders. 2019. "Mixed-Precision Analysis of Householder QR Factorization." Submitted to *SIAM Journal on Scientific Computing*. LLNL-JRNL-795525-DRAFT.

Zhang, X., J.-Y. Trepanier, and R. Camarero. 2000. "A posteriori error estimation for finite-volume solutions of hyperbolic conservation laws." *Computer Methods in Applied Mechanics and Engineering* 185(1):1-19. doi:10.1016/S0045-7825(99)00099-7.

Publications and Presentations

Peer-reviewed Publications

Diffenderfer, J., A. Fox, J. Hittinger, G. Sanders, and P. Lindstrom. 2019. "Error Analysis of ZFP Compression for Floating-point Data," *SIAM Journal on Scientific Computing* 41(3):A1867-A1898. LLNL-JRNL-744818. doi:10.1137/18M1168832.

Fox, A., J. Diffenderfer, J. Hittinger, G. Sanders, and P. Lindstrom. 2019. "Stability Analysis of Inline ZFP Compression for Floating-point Data in Iterative Methods." *SIAM Journal on Scientific Computing*. Accepted with minor revisions, October 2019. LLNL-JRNL-769679-DRAFT.

Fox, A., G. Sanders, and A. Knyazev. 2018. "Investigation of Spectral Clustering for Signed Graph Matrix Representation." *IEEE HPEC Conference Proceedings*, September 2018. LLNL-CONF-754816. doi:10.1109/HPEC.2018.8547575.

Hammerling, D., A. Baker, A. Pinard, and P. Lindstrom. 2019. "A Collaborative Effort to Improve Lossy Compression Methods for Climate Data." 5th International Workshop on Data Analysis and Reduction for Big Scientific Data, in conjunction with SC19, to appear November 2019. LLNL-CONF-790800.

Hoang, D., P. Klacansky, H. Bhatia, P.-T. Bremer, P. Lindstrom, and V. Pascucci. 2019. "A Study of the Trade-off Between Reducing Precision and Reducing Resolution for Data Analysis and Visualization." *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1193-1203. LLNL-CONF-755005. doi:10.1109/TVCG.2018.2864853.

Lam, M., T. Vanderbruggen, H. Menon, and M. Schordan. 2019. "Tool Integration for Source-Level Mixed Precision." Correctness'19: Third International Workshop on Software Correctness for HPC Applications, in conjunction with SC19, to appear Nov 2019. LLNL-CONF-787885.

Lindstrom, P., S. Lloyd, and J. Hittinger. 2018. "Universal Coding of the Reals: Alternatives to IEEE Floating Point." In Proceedings of the Conference for Next Generation Arithmetic 2018, Singapore, 28 March 2018. In cooperation with ACM. LLNL-CONF-743265. doi:10.1145/3190339.3190344. ***Deputy Director for Science and Technology Excellence in Publication Award.***

Lindstrom, P. 2019. "Universal Coding of the Reals using Bisection." In Proceedings of the Conference for Next Generation Arithmetic 2019, Singapore, 13 March 2019. In cooperation with ACM. LLNL-CONF-764361. doi:10.1145/3316279.3316286.

Menon, H., M. Lam, D. Osei-Kuffuor, M. Schordan, S. Lloyd, K. Mohror, and J. Hittinger. 2018. "ADAPT: Algorithmic differentiation applied to floating-point precision tuning." In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, 11-16 November 2018, Paper 48:1–13. LLNL-CONF-748742. doi:10.1109/SC.2018.00051.

Schordan, M., J. Huckelheim, P.-H. Lin, and H. Menon. 2017. "Verifying the Floating-Point Computation Equivalence of Manually and Automatically Differentiated Code." Correctness'17: First International Workshop on Software Correctness for HPC Applications, in conjunction with SC17, Denver, Colorado, 12 November 2017. In Proceedings of the First International Workshop on Software Correctness for HPC Applications (Correctness'17), New York, NY: ACM, 34-41. LLNL-CONF-737605. doi:10.1145/3145344.3145489.

Non-Peer Reviewed Publications

Hittinger, J. 2019. "Variable Precision Computing Improves Simulation Speed and Efficiency." Computing Report, Lawrence Livermore National Laboratory, p. 42-45. LLNL-TR-787575.

Lindstrom, P. 2017. "Error Distributions of Lossy Floating-Point Compressors." Proceedings of 2017 Joint Statistical Meetings, Baltimore, MD, 29 July – 3 August 2017. LLNL-CONF-740547.

Metere, A. and G. Friedland. 2018. "Isomorphism between the Maximum Lyapunov Exponent and the Shannon's Channel Capacity." LLNL-TR-733786.

Publications in Review

Yang, L., A. Fox, and G. Sanders. 2019. "Mixed-Precision Analysis of Householder QR Factorization." Submitted to *SIAM Journal on Scientific Computing*. LLNL-JRNL-795525-DRAFT.

Publications in Preparation

Bhatia, H., D. Hoang, G. Morrison, V. Pascucci, P.-T. Bremer, and P. Lindstrom. 2019. "Mixed-Precision Adaptive Multilinear Meshes." *IEEE Transactions on Visualization and Computer Graphics*. Planned submission January 2020. LLNL-CONF-771697-DRAFT.

Copeland, D., D. Osei-Kuffuor, and J. Hittinger. 2019. "Roundoff Error Correction for Parabolic PDEs Simulated in Mixed Precision using Error Transport." *SIAM Journal on Scientific Computing*. Planned submission December 2019.

Fox, A. and P. Lindstrom. "Statistical Analysis of ZFP: Understanding Lossy Compression Error Bias." *SIAM Journal on Scientific Computing*. Planned submission December 2019.

Metere, Alfredo. "Detecting rare events in molecular dynamics simulation." In preparation.

Invited, Plenary, and Keynote Presentations

Hittinger, J. "Making Every Bit Count: Variable Precision?" Big Data Meets Computation Workshop, Institute for Pure and Applied Mathematics, UCLA, Los Angeles, CA, Feb 3, 2017. LLNL-PRES-722917.

Hittinger, J. "Variable Precision: Making Every Bit Count?", Computer Science and Engineering Department Seminar, Lehigh University, Bethlehem, PA, Oct 19, 2017. LLNL-PRES-722917.

Hittinger, J. "Variable Precision: Making Every Bit Count." Université Nice Sophia Antipolis, Nice, France, June 25, 2018. LLNL-PRES-753450.

Hittinger, J. "Variable Precision: Making Every Bit Count." PASC18, Basel, Switzerland, July 3, 2018. LLNL-PRES-753451.

Hittinger, J. "Variable Precision Computing Research in the Center for Applied Scientific Computing." Technische Universität München, Garching, Germany, July 6, 2018. LLNL-PRES-753448.

Hittinger, J. "Rethinking Finite Precision." Science at Extreme Scales Workshop IV. Institute for Pure and Applied Mathematics, UCLA, Los Angeles, November 20, 2018. LLNL-PRES-767245.

Hittinger, J. "Variable Precision Computing Research in the Center for Applied Scientific Computing." University of Illinois Urbana-Champaign Computational Science and Engineering Seminar, Urbana, IL, April 16, 2019. LLNL-PRES-753448.

Hittinger, J. "Variable Precision Computing Strategies." ISC High Performance 2019, Frankfurt, Germany, Jun 17, 2019. LLNL-PRES-779040.

Lindstrom, P. "Lossy Compression of Floating-Point Data." International Computing for the Atmospheric Sciences Symposium, Annecy, France, September 10-14, 2017. LLNL-PRES-739189.

Lindstrom, P. "Universal Coding of the Reals: Alternatives to IEEE Floating Point." Keynote, Conference for Next Generation Arithmetic 2018, Singapore. March 28, 2018. LLNL-PRES-748385.

Lindstrom, P. "Alternatives to IEEE: NextGen Number Formats for Scientific Computing." Science at Extreme Scales Workshop. Institute for Pure and Applied Mathematics, UCLA, Los Angeles, October 15, 2018. LLNL-PRES-759606.

Menon, H. "Error Analysis in HPC Applications Using Algorithmic Differentiation." James Madison University, Harrisonburg, VA, April 3, 2019. LLNL-PRES-759649.

Osei-Kuffuor, D. "Variable Precision Computing for Scientific Applications." Smoky Mountains Conference on Computational Science and Engineering, Kingsport, TN, 27-29 August 2019. LLNL-PRES-788465.

Contributed Presentations

Diffenderfer, J. "Error Analysis of ZFP Compression for Floating-Point Data." SIAM Conference on Computational Science and Engineering, Spokane, WA, February 28, 2019. LLNL-PRES-768145.

Fox, A. "Propagation of Compression Error of ZFP with Fixed Precision for Floating-Point Data in Iterative Methods." CM2018: Fifteenth Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, March 30, 2018. LLNL-PRES-757672.

Fox, A. "Stability Analysis of Inline ZFP Compression for Floating-point Data in Iterative Methods." SIAM Conference on Computational Science and Engineering, Spokane, WA. February 28, 2019. LLNL-PRES-768209.

Fox, A. "Error Analysis of Inline ZFP Compression for Multigrid Methods." Copper Mountain Conference Iterative Methods, Copper Mountain, CO, March 25, 2019. LLNL-PRES-770240.

Hittinger, J., P. Lindstrom, and S. Lloyd. "Universal Coding of the Reals: Alternatives to IEEE Floating Point -An Example Application." SC17 BoF: Improving Numerical Computation with Practical Tools and Novel Computer Arithmetic, Denver, CO, November 14, 2017. LLNL-PRES-741490.

Hittinger, J., P. Lindstrom, D. Osei-Kuffuor, A. Fox, G. Sanders, S. Lloyd, J. Diffenderfer, and J. Gordon. "Adapting Precision in Scientific Simulations." 13th World Congress on Computational Mechanics, New York, NY, July 26, 2018. LLNL-PRES-756553.

Hoang, D. "A Study of the Trade-off between Reducing Precision and Reducing Resolution for Data Analysis and Visualization." IEEE VIS Berlin, Germany, October 2018.

Kolasinksi, A. "Error Analysis of Inline ZFP Compression for Multigrid Methods." SIAM Conference on Computational Science and Engineering, Spokane, WA, February 28, 2019. LLNL-PRES-770240.

Lindstrom, P. "Lossy Compression Algorithms for Floating-Point Data." Joint Statistical Meetings, Baltimore, MD., July 29-August 3, 2017. LLNL-PRES-736379.

Lindstrom, P. "Universal Coding of the Reals using Bisection." SC17 BoF: Improving Numerical Computation with Practical Tools and Novel Computer Arithmetic, Denver, CO, November 14, 2017. LLNL-PRES-731413.

Lindstrom, P. "Beyond IEEE: Next-Generation Floating-Point Formats." SIAM Conference on Computational Science and Engineering, Spokane, WA, February 28, 2019. LLNL-PRES-768285.

Menon, H. "ADAPT: Algorithmic Differentiation Applied to Floating-Point Precision Tuning." International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, 14 November 2018. LLNL-PRES-759649.

Menon, H. "Error Analysis in HPC Applications Using Algorithmic Differentiation." Ninth International Women in HPC Workshop, in conjunction with SC18, Dallas, TX, November 11, 2018. LLNL-PRES-759413.

Sanders, G., A. Fox, J. Herring, J. Hittinger, and D. Osei-Kuffuor. "Floating Point Analysis for Half-Precision Eigensolvers." CM2018: Fifteenth Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, March 30, 2018. LLNL-PRES-748762. Schordan, M. "Verifying the Floating-Point Computation Equivalence of Manually and Automatically Differentiated Code." Correctness'17: First International Workshop on Software Correctness for HPC Applications, Denver, CO, November 12, 2017. LLNL-PRES-741134.

Yang, M. "Half-Precision Block QR Factorization." SIAM Conference on Computational Science and Engineering, Spokane, WA, February 28, 2019. LLNL-PRES-768669.

Poster Presentations

Diffenderfer, J. "Implementing Anderson Acceleration in Mixed Precision." LLNL Summer Student Poster Symposium, August 2, 2017. LLNL-POST-735360.

Fox, A., J. Diffenderfer, J. Hittinger, G. Sanders, and P. Lindstrom. "Stability Analysis of inline ZFP Compression for Floating-Point Data in Iterative Methods." 25th International Domain Decomposition Conference, Newfoundland, Canada. July 27, 2018. LLNL-POST-752539. ***Best Poster Award.***

Fox, A., G. Sanders, and A. Knyazev. "Investigation of Spectral Clustering for Signed Graph Matrix Representation." IEEE HPEC Conference, September 25, 2018. LLNL-POST-758625.

Gordon, J. "Analysis of Mixed Precision Iterative Refinement Method." LLNL Summer Student Poster Symposium, August 2, 2017. LLNL-POST-735423-DRAFT.

Herring, J. "Variable Precision: toward a GPU enabled LOBPCG in HYPRE." LLNL Summer Student Poster Symposium, August 2, 2017. LLNL-POST-735534.

Hoang, D., and P. Lindstrom. "Compression of particle simulation data using space partitioning and binomial coding." LLNL Student Poster Symposium, August 2018. LLNL-POST-755761.

Kolasinksi, A., and A. Fox. "Error Analysis of Inline ZFP Compression for Multigrid Methods." LLNL Student Poster Symposium, August 2018. LLNL-POST-755648.

Lindstrom, P. "Reducing Data Movement using Adaptive-Rate Computing." Computation Directorate External Review Committee Meeting, LLNL, Livermore, CA, May 9-11, 2017. LLNL-POST-728998.

Moody, L., N. Pinnow, M. Lam, H. Menon, M. Schordan, S. Lloyd, and T. Islam. "Automatic Generation of Mixed-Precision Programs." International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, November 11-16, 2018. LLNL-POST-756004.

Moody, L., S. Lloyd, and P. Lindstrom. "Viability of Hardware Accelerated Floating-Point Compression", LLNL Summer Student Poster Symposium, August 2019, LLNL-POST-783177.

Usher, W., D. Hoang, V. Pascucci, P. Lindstrom, "Exploring ZFP for Compression of Particle Data." LLNL Student Poster Symposium, August 2018. LLNL-POST-755563.

Yang, L. M., and G. D. Sanders. "Simulated Half-Precision Implementation of Blocked QR Factorization and Graph Clustering Applications." LLNL Student Poster Symposium, August 2018. LLNL-TR-756282. ***Top 10% Poster Symposium Prize.***

Open Source Software

- ADAPT: https://github.com/LLNL/adapt-fp
- FloatSmith: https://github.com/crafthpc/floatsmith
- MAMM: http://github.com/LLNL/AMM
- Typeforge: https://github.com/rose-compiler/rose/tree/release/projects/typeforge
- ZFP: https://github.com/LLNL/zfp

Notes to the Editors

- Figure 3 includes superscripts (10 to the -2, 10 to the -1, 10 to the -6, 10 to the -9, 10 to the -12, 10 to the -15)