

Optimizing Next Generation Hydrodynamics Code for Exascale Systems

Dana Akhmetova
KTH Royal Institute of Technology
Stockholm, Sweden
danaak@kth.se

Frederick Ouellet
University of Florida
Gainesville, Florida
f.ouellet@ufl.edu

Christoph Junghans
Los Alamos National Laboratory
Los Alamos, New Mexico
junghans@lanl.gov

Sumathi Lakshmiranganatha
University of Wyoming
Laramie, WY
slakshmi@uwyo.edu

Patrick Payne
Los Alamos National Laboratory
Los Alamos, New Mexico
ppayne@lanl.gov

Robert Pavel
Los Alamos National Laboratory
Los Alamos, New Mexico
rspavel@lanl.gov

Diptajyoti Mukherjee
Allegheny College
Meadville, Pennsylvania
mukherjeed2@allegheny.edu

Nicholas Stegmeier
South Dakota State University
Brookings, South Dakota
nicholas.stegmeier@sdstate.edu

Vinay Ramakrishnaiah
Los Alamos National Laboratory
Los Alamos, New Mexico
vinayr@lanl.gov

ABSTRACT

Studying continuum dynamics problems computationally can illuminate complex physical phenomena where experimentation is too costly. However, the models used in studying these phenomena usually require intensive calculations, some of which are beyond even the largest supercomputers to date. Emerging high performance computing (HPC) platforms will likely have varied levels of heterogeneity, making hybrid programming with MPI+X essential for achieving optimal performance. This research investigates hybrid programming and unconventional approaches like machine learning for a next generation hydrodynamics code, FleCSALE, in the context of tabular equation of state (EOS). We demonstrate an overall 5x speedup to the code, the use of GPUs to accelerate EOS tabular interpolation, and a proof of concept machine learning approach to EOS.

CCS CONCEPTS

• **Computing methodologies** → *Parallel computing methodologies; Distributed computing methodologies*; Machine learning; Modeling and simulation;

KEYWORDS

Equation of State, MPI, GPU, OpenMP, Machine Learning, FleCSALE, FleCSI

ACM Reference Format:

Dana Akhmetova, Sumathi Lakshmiranganatha, Diptajyoti Mukherjee, Frederick Ouellet, Patrick Payne, Nicholas Stegmeier, Christoph Junghans, Robert

Pavel, and Vinay Ramakrishnaiah. 2018. Optimizing Next Generation Hydrodynamics Code for Exascale Systems. In *Proceedings of Supercomputing Conference (SC18)*. ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

FleCSALE is a software package developed for studying continuum dynamics problems, such as fluid flow [2]. It is specifically developed for existing and emerging large distributed memory system architectures. These emerging high performance computing (HPC) platforms will likely have varied levels of heterogeneity. As such, reaching peak performance will require combined distributed and shared memory models, such as MPI+X, with full awareness of hardware aspects like memory locality and compute and message passing concurrency. To accommodate future hardware architectures, FleCSALE is built on FleCSI, the Flexible Computational Science Infrastructure, which is designed to allow flexibility in choosing runtime implementations and optimizations [1].

The goal of this research was to increase the performance of FleCSALE for hybrid platforms like Sierra and Summit, particularly in the context of tabular equation of state (EOS) through the EOSPAC library [5]. The computationally intensive portions of FleCSALE, namely the calls to EOSPAC interpolation algorithms, were targeted for GPU offloading due to their comparatively static input and parallelizability. On the other hand, the less compute intensive portions of FleCSALE were implemented with OpenMP to demonstrate fine-grained parallelism within the FleCSI framework. These two approaches were pursued with the eventual goal of using task based parallelism to employ diverse types of compute resources simultaneously. Machine learning (ML) was investigated as a way to bypass the computationally expensive interpolation calls to EOSPAC and reduce the memory required to load the material data tables.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SC18, November 2018, Dallas, Texas

© 2016 Copyright held by the owner/author(s).

ACM ISBN 123-4567-24-567/08/06.

https://doi.org/10.475/123_4

LA-UR-18-27448

2 METHODOLOGY

2.1 EOSPAC Interface Optimizations

An area of optimization for the overall code was the interfacing between FleCSALE and its equation of state library, EOSPAC. This library computes the pressure, temperature and speed of sound from density and internal energy by looking up material-specific data tables from a linked database and interpolating the needed values. However, these tables are written such that the library inverts the data tables each time they are used and then removes them from memory. The use of EOSPAC within FleCSALE was modified to find which tables are required during runtime and perform any necessary inversions at initialization and store them into memory [5]. A second improvement was done by sending all interpolation points for a given material type which are controlled by a single processor to the interpolation routines as a block instead of the original loop over individual interpolation points. A final optimization was to modify the way in which EOSPAC searched through the data tables to place the interpolation stencil. By presorting the input values to the search function, the interpolation stencil could be placed more efficiently.

2.2 GPU Porting for EOSPAC

On profiling FleCSALE, we found that EOSPAC interpolation calls were the costliest part of FleCSALE, computationally. Interpolation algorithms are generally parallelizable and compute intensive rather than memory intensive, and as such are suitable for porting to GPGPUs. We ported the EOSPAC birational [3] and bilinear interpolation routines to GPGPUs using CUDA to investigate potential performance improvements within EOSPAC and FleCSALE. Several memory management schemes were explored, including synchronous memory copies, CUDA unified memory, host pinned memory, and texture memory. CUDA was chosen as it offered greater control over these memory management schemes, compared to other GPU parallel computing platforms.

2.3 Machine Learning for EOS

We investigated the viability of algorithms in the DLIB C++ ML library [4] to replace costly EOSPAC queries in FleCSALE. Kernel ridge regression (KRR) and random forest (RF) regression models were pre-trained using EOSPAC interpolated data and loaded within a FleCSALE simulation. Machine learning for thermodynamic equation of state has several challenges, including maintaining thermodynamic consistency and monotonicity constraints. Despite these challenges, the KRR and RF models reproduced the expected simulation results within FleCSALE, demonstrating a proof of concept machine learning approach to EOS. Memory, efficiency, and accuracy trade-offs are compared between traditional tabular EOS and ML EOS.

2.4 Hybrid Programming with OpenMP

OpenMP, the de-facto standard for shared-memory programming, was used in outer for-loop work-sharing constructs in FleCSALE tasks related to solving full hydro solutions (setting initial conditions, normalization of materials, updating material states, computing nodal quantities, saving mesh coordinates, etc). It should be

noted that OpenMP was not integrated with the EOSPAC library but only with FleCSALE.

2.5 Sparse Data Optimization

FleCSI uses MPI One-sided communication as a distributed memory backend model and FleCSALE uses sparse data [2]. Tracing the application on a node with more than one MPI process demonstrated huge communication latency with `MPI_Win_create` being the most expensive call. We analyzed in the source code, eliminated the unnecessary copies and shared window creation which showed on-node and multi-node performance improvements. By packing the data into a single message using `MPI_Datatype`, we were able to achieve significant improvements in the MPI communication and were able to obtain better scalability, as shown in Figure 1.

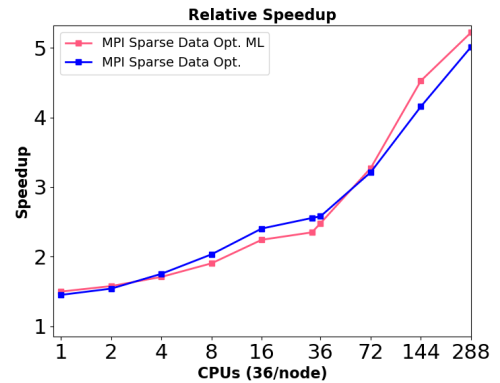


Figure 1: Strong scaling results showing the original code, the sparse data optimizations, and the integrated machine learning models.

3 CONCLUSIONS

This research investigated optimization strategies for FleCSALE in the context of tabular equation of state. Sparse data optimization and batch interpolation processing proved successful and gave a speedup of 5x, and presorting interpolation variables before querying EOSPAC could provide further speedup depending on the simulation context. Hybrid programming with OpenMP demonstrates that fine grained parallelism can be achieved within FleCSALE. Porting EOSPAC interpolation calls to GPUs is also promising, and work is ongoing to integrate the GPU interpolation kernels with FleCSALE. Although further optimization is needed, machine learning was used to model EOS within FleCSALE with sufficient accuracy, showing improved performance in the case of KRR models.

ACKNOWLEDGMENTS

This research is funded by Los Alamos National Laboratory, the Information Science and Technology Institute, and the Advanced Simulation and Computing Program.

REFERENCES

- [1] Ben Bergen, Nicholas Moss, and Marc Robert Joseph Charest. 2016. Flexible Computer Science Infrastructure (FleCSI), Version 00. <https://www.osti.gov/servlets/purl/1311634>
- [2] Aimee L. Hungerford and David John Daniel. [n. d.]. (U) Ristra Next Generation Code Report. ([n. d.]). <https://doi.org/10.2172/1394959>
- [3] G. I. (Gerald I.) Kerley. 1977. *Rational function method of interpolation*. Technical Report.
- [4] Davis E. King. 2009. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research* 10 (2009), 1755–1758.
- [5] David A. Pimentel and Daniel G. Sheppard. [n. d.]. Pre-inverted SESAME data table construction enhancements to correct unexpected inverse interpolation pathologies in EOSPAC 6. ([n. d.]). <https://doi.org/10.2172/1419738>