Enabling code portability of a parallel and distributed smooth-particle hydrodynamics application, FleCSPH

Suyash Tandon[1], Nicholas Stegmeier[2], Vasu Jaganath[3], Jennifer Ranta[4], Rathish Ratnasingam[5], Elizabeth Carlson[6], Julien Loiseau[7], Vinay Ramakrishnaiah[7] and Robert Pavel [7]
[1]University of Michigan, [2]University of Illinois, [3]University of Wyoming, [4]Michigan State University, [5]Newcastle University, [6]University of Nebraska-Lincoln, [7] Los Alamos National Laboratory
suyashtn@umich.edu,nws4@illinois.edu,vjagana1@uwyo.edu,jloiseau@lanl.gov

ABSTRACT

Core-collapse supernovae (CCSNe) are integral to the formation and distribution of heavy elements across the universe. However, CCSNe are highly complex and inherently non-linear phenomena. Large-scale simulations of these cosmic events can provide us a glimpse of their hydrodynamic and nucleosynthetic processes which are difficult to observe. To enable these massive numerical simulations on high-performance computing (HPC) centers, this study uses FleCSPH, a parallel and distributed code, based on the smooth-particle hydrodynamics (SPH) formulation. In the recent years, the HPC architecture has evolved and the next generation of exascale computers are expected to feature heterogenous architecture. Therefore, it is important to maintain code portability across platforms. This work demonstrates code portability of FleCSPH through the incorporation of Kokkos C++ library and containers using Charliecloud.

KEYWORDS

Performance portability, containers, Kokkos, Charliecloud, applications

ACM Reference Format:
Suyash Tandon[1], Nicholas Stegmeier[2], Vasu Jaganath[3], Jennifer Ranta[4], Rathish Ratnasingam[5], Elizabeth Carlson[6], Julien Loiseau[7], Vinay Ramakrishnaiah[7] and Robert Pavel [7]. 2019. Enabling code portability of a parallel and distributed smooth-particle hydrodynamics application, FleCSPH. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage and Analysis (SC19).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Core-collapse supernovae (CCSNe) – explosions of stars more than ten times as massive as the sun – are nature's grandest explosions and astrophysical laboratories where conditions develop that are not achievable on Earth. It is in these furnaces that many of the elements heavier than carbon are forged [6]. The rise in the computing technology in the last two decades, has enabled scientists to study the complex and nonlinear physics of CCSNe and the resulting production and distribution of their nucleosynthetic yields through numerical simulations. The challenges in these time-evolution simulations arise from: 1) multi-scale nature, as, the spatial length and density scales can span as much as ten orders of magnitude [4]. Such resolution requirements can be quite daunting and computationally expensive; and 2) the CCSNe are rich in physics and generally involve forces and interactions from gravity, fluid instabilities, nuclear reactions, and much more. To enable these massive numerical simulations on high-performance computing (HPC) centers, this study uses FleCSPH, a parallel and distributed code, based on the smooth-particle hydrodynamics (SPH) formulation [2].

In the recent years, the HPC architecture has evolved and the next generation of exascale computers are expected to feature heterogenous architecture. Therefore, it is important to maintain code portability across platforms. Performance portability of FleCSPH is facilitated through the incorporation of Kokkos framework [1] with multiple backend support for multi-CPU and/or multi-GPU systems. Furthermore, this work investigates the use of Charliecloud [5] to build containers for user-defined software stack (UDSS) of FleCSPH that avoids most security risks without compromising the performance and functionality provided by the underlying hardware.

2 CCSNE SIMULATIONS WITH FLECSPH

FleCSPH [3], a smoothed particle hydrodynamics (SPH) implementation, is an explicit, mesh-free, numerical method that solves hydrodynamic equations by discretizing the partial differential equations as a set of fluid elements called particles.

The parallel and distributed structure of FleCSPH relies on the MPI+X framework, where inter-node parallelization is enabled through MPI and intra-node parallelism is enabled by using thread-level parallelism such OpenMP. The particles in the computational domain in FleCSPH, are divided roughly equally between available MPI ranks. Then each MPI rank, builds a n−dimensional (Oct-tree for $n = 3$) tree-topology, fig.1. The tree traversal and particle interactions inside each MPI rank, was originally handled using OpenMP pragmas. The boundary particles in each MPI rank are exchanged with other MPI ranks after every iteration. Thus, of the overall compute time per iteration, 40% of the time is spent in tree traversal and particle-interactions.

To simulate CCSNe with FleCSPH, a number of algorithmic extensions were added including model improvement with additional terms in SPH formulation for better particle approximations that added complexity of *O(N)*, and investigation of suitable parameters to accelerate N-body gravitational interaction to *O(N log N)*. Other

SC19, November 17-22, 2019, Denver, CO

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

LA-UR-19-27973

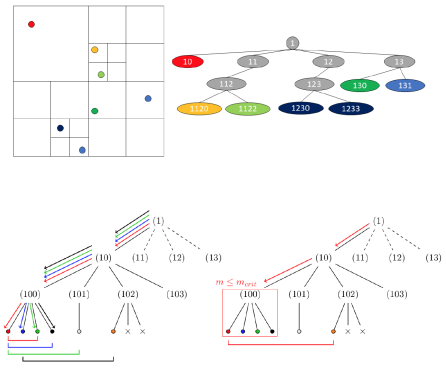

Figure 1: Tree topology and tree traversal in FleCSPH [3].

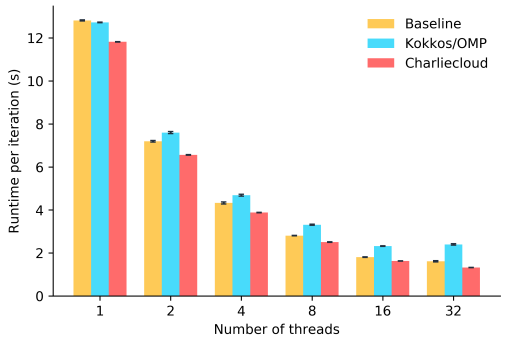

Figure 2: Code portability of FleCSPH using Kokkos and Charliecloud.

additions included physics models, and tabulated data specific for CCSNe. These models and improvements were tested and validated with a benchmark white-dwarf.

3 CODE PORTABILITY

Maintaining code portability is essential to facilitate large-scale simulations across different HPC platforms. FleCSPH uses the Kokkos C++ library and especially targets the intra-node, thread-level parallelism of the tree traversal described earlier. For this purpose, the standard C++ data structure such as vectors and unordered maps were ported to Kokkos supported data structures such as Views. The OpenMP pragma constructs were replaced with Kokkos parallel dispatch loops with lamdas enabled. At present FleCSPH features Kokkos OpenMP backend support. Kokkos CUDA backend support needs special attention, in particular due to the C++ 17 conformance standards of FleCSPH. In the most recent build, FleCSPH was downgraded to C++14 for Kokkos CUDA support, but needs further testing and validation to ensure stability of the application.

A physics based benchmark test, the 3D Sod Shock Tube, was conducted with increasing number of thread count on one node on Intel broadwell $E5-2695\_v4$ to show the comparison of the thread-level parallelism in FleCSPH between the baseline and Kokkos

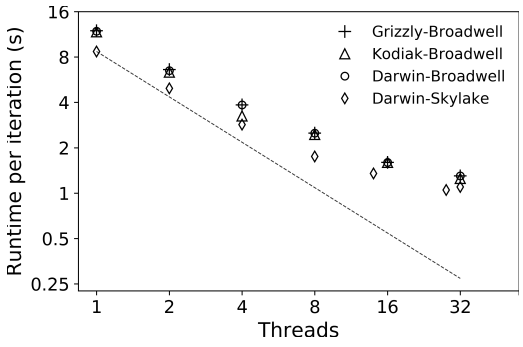

Figure 3: Code portability of FleCSPH on HPC platforms.

OpenMP backend versions, see fig. 2. For the test, number of particles were fixed as $N = 0.5$ million and iteration count was 20. As thread count increases, the time per iteration reduces, however, performance of Kokkos OpenMP backend is slower than baseline by approximately 20% which can be attributed to the modifications in data structures. 5 sample runs for each thread count for each version were conducted. The standard deviation was recorded as 0.027 for baseline and 0.032 for Kokkos OpenMP backend, and is plotted as error bars in fig. 2.

Furthermore, a software stack of FleCSPH was created in a container using Charliecloud. Here, the baseline version of FleCSPH is encapsulated in the container and benchmark test similar to that of Kokkos OpenMP backend is conducted. Launching the application inside the container witnesses a tiny performance gain, as seen in fig. 2. This is possible due to multiple factors including lower latency of I/O operations inside the container, the image being loaded into the RAM, and therefore, necessitates further investigation. It must be noted that the baseline and Charliecloud image were run on the same node to ensure similarities in the hardware resources on the host and the containers for performance measurements. Fig. 3 shows stack portability of the same container image on Grizzly and Kodiak on Intel broadwell $E5-2695\_v4$ nodes and the Darwin testbed HPC cluster on Intel broadwell $E5-2695\_v4$ and skylake nodes.

4 CONCLUSIONS

In the poster session, we intend to present the improved version of FleCSPH and the test case results. The thread-level parallelism using Kokkos and its affect on performance portability will be demonstrated. The SPH simulations of core-collapse supernovae will be discussed with an emphasis on the importance of HPC for such multi-physics astrophysical problems. Finally, we will also discuss the concept of containerization of HPC applications and its effect on code portability across platforms.

ACKNOWLEDGMENTS

This study is funded by Los Alamos National Laboratory, the Information Science & Technology Institute and the Advanced Simulation and Computing Program. We greatly benefitted from the support of LANL's HPC Division and the Darwin Admin Team.

LA-UR-19-27973

REFERENCES

[1] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. 2014. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *J. Parallel and Distrib. Comput.* 74, 12 (2014), 3202 – 3216. https://doi.org/10.1016/j.jpdc.2014.07.003 Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[2] MB Liu and GR Liu. 2010. Smoothed particle hydrodynamics (SPH): an overview and recent developments. *Archives of computational methods in engineering* 17, 1 (2010), 25–76.

[3] Julien Loiseau, Hyun Lim, Ben K Bergen, Nicholas D Moss, and Francois Alin. 2018. FleCSPH: a Parallel and Distributed Smoothed Particle Hydrodynamics Framework Based on FleCSI. *2018 International Conference on High Performance Computing & Simulation (HPCS)* (2018), 484–491.

[4] Christian D Ott. 2016. Massive Computation for Understanding Core-Collapse Supernova Explosions. *Computing in Science & Engineering* 18, 5 (2016), 78–92.

[5] Reid Priedhorsky and Tim Randles. 2017. Charliecloud: Unprivileged containers for user-defined software stacks in hpc. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2017), 36.

[6] Stan Woosley and Thomas Janka. 2005. The physics of core-collapse supernovae. *Nature Physics* 1, 3 (2005), 147.

LA-UR-19-27973