Towards an Exascale-Ready Mini-App for Smooth Particle Hydrodynamics

Florina M. Ciorba‡, Lucio Mayer‡, Rubén Cabezon‡, David Imbert†, Danilo Guerrera†, Aurélien Cavelan†, Darren S. Reed‡, Jean-Guillaume Piccini‡, Ioana Banicescu‡, Domingo García-Senz‡, and Thomas R. Quinn†

‡ University of Basel, † University of Zürich, ‡ Nextflow Software, † Swiss National Supercomputing Centre (CSCS), ‡ Mississippi State University, † University of Washington in Seattle, ‡ Universitat Politècnica de Catalunya

1. Vision

- Assess the performance of state-of-the-art Smooth Particle Hydrodynamics codes
- Understand their implementation limits
- Design a mini-app that synthesizes their characteristics
- Design parallelization methods and fault-tolerance mechanisms to sustain their scalability on massively parallel environments
- Employ state-of-the-art dynamic load balancing to address load imbalance

2. Smooth Particle Hydrodynamics (SPH)

- Purely Lagrangian method, with no subjacent mesh
- Used in many numerical simulations, including astrophysics and computational fluid dynamics
- Discretizes the fluid in a series of interpolation points (SPH particles) distributed to follow the mass density of the gas or fluid
- Particle simulation interpolates over close neighboring particles with various weights
- Computationally demanding in terms of sustained floating point operations per second
- Codes studied in PASC SPH-EXA [1]: SPHYNX [2], ChaNGa [3] and SPH-flow [4, 5]

3. Test Cases

Rotating Square Patch [6]

- Standard test for computational fluid dynamics simulations
- Rotation of a free-surface square fluid patch
- Performed in 3D with 10^6 particles
- 20 simulation time-steps

Evrard Collapse [7]

- Standard test for astrophysics simulations with self-gravity
- Adiabatic collapse of an initially cold and static gas sphere
- Performed in 3D with 10^6 particles
- 20 simulation time-steps

4. Experimental Results

5. Reproducibility Setup

The 3 SPH codes test cases executed on MareNostrum [3] and Piz Daint [6]. Here is a description of the environments and configurations used for the experiments:

7. Acknowledgments

This work is supported by the Swiss Platform for Advanced Scientific Computing (PASC) project SPH-EXA [1] (2017-2020). The authors acknowledge the support of the Swiss National Supercomputing Centre (CSCS) via allocation project c16, Barcelona Supercomputing Center (BSC), and the Center for scientific computing at the University of Basel (scienceCORE) where the calculations were performed. Some of the computation and simulation results presented in this document were obtained with SPH-flow, courtesy of Nextflow Software.

6. SPH-EXA Mini-App

- Extrapolates common features of state-of-the-art SPH codes
- Addresses the limits of the actual implementations of SPHYNX, ChaNGa and SPH-flow
- Provides a reference optimized implementation (MPI+X) in C++
- Employs state-of-the-art load balancing and fault-tolerance techniques
- Provides an optimized implementation of basic SPH operators
- Porting the mini-app to new programming paradigms (e.g., HPX)
- Verification and reproducibility of the SPH simulations
- Validation of the new implementation against the parent codes

8. References