Porting our astrophysics application to Arm64FX and adding Arm64FX support using kokkos

Patrick Diehl and Gregor Daisß

Louisiana State University and University of Stuttgart

patrickdiehl@lsu.edu and Gregor.Daiss@ipvs.uni-stuttgart.de

02/10/2022
Motivation

At peak brightness, the rare 2002 red nova V838 Monocerotis briefly rivalled the most powerful stars in the Galaxy. Credit: NASA/ESA/H. E. Bond (STScI)

Goal

Simulate the merger and obtain the light curve to understand the observations better:

**Multi- physic** is need:
- Hydro
- Gravity
- Radiation

Reference

Overview

1. Software framework
   - Octo-Tiger
   - HPX
   - Kokkos and HPX

2. Scaling Results on Ookami

3. Conclusion and Outlook
Octo-Tiger

Astrophysics open source program simulating the evolution of star systems based on the fast multipole method on adaptive Octrees.

Modules
- Hydro
- Gravity
- Radiation (benchmarking)

Supports
- Communication: MPI/libfabric
- Backends: CUDA, HIP, Kokkos

Reference

https://github.com/STELLAR-GROUP/octotiger
Example of a merger simulation

Figure 2. The early stages of mass transfer in a binary star system. The accreting star is five times more massive than the donor star.

Reference

HPX

HPX is an open source C++ Standard Library for Concurrency and Parallelism\(^2\).

**Features**

- HPX exposes a uniform, standards-oriented API for ease of programming parallel and distributed applications.
- HPX provides unified syntax and semantics for local and remote operations.
- HPX exposes a uniform, flexible, and extendable performance counter framework which can enable runtime adaptivity.

**Reference**


\(^2\)https://github.com/STEllA-HPC/HPX
HPX-Kokkos in Octo-Tiger

- HPX: Combine Tasks via futures in different ways (DAG)
- Octo-Tiger: Uses Kokkos for compute-intensive kernels (running on CPU+GPU)
- Uses HPX-Kokkos integrations (futurized Kokkos kernels to integrate into the dependency graph + kernels as tasks)
- Octo-Tiger’s Kokkos kernels use Kokkos SIMD types for explicit vectorization

Reference

Scaling Results on Ookami
Test Setup

Purpose of the Test:
- Basic check for (node-level, distributed) scaling with HPX on ARM
- Basic check for SIMD speedup using the Kokkos SIMD (NEON) types
Test Setup

Purpose of the Test:
- Basic check for (node-level, distributed) scaling with HPX on ARM
- Basic check for SIMD speedup using the Kokkos SIMD (NEON) types

Setup:
- Gravity-only scenario ("solid sphere") using the Fast Multipole Method (FMM)
- Runtime averages over 10 FMM iterations
- Level 5 AMR oct-tree, each node has one sub-grid of the size 8x8x8
- Overall, using 7625 Subgrids with 512 cells each

→ Small scenario that can still be run on one core for scaling tests
Node-Level on 1 ARM64FX Node

Node-level Scaling of the FMM Gravity solver
(SCALAR -> NEON Speedup: 1.4)

![Graph showing the scaling of the FMM Gravity solver with NEON and SCALAR implementations](image)

- NEON: FMM Iteration Runtime (s)
- NEON: Parallel Efficiency
- SCALAR: FMM Iteration Runtime (s)
- SCALAR: Parallel Efficiency
Distributed on 1 to 8 Nodes

Distributed Scaling of the NEON build

MPI backend: FMM Iteration Runtime (s)
MPI backend: Parallel Efficiency

Number of Cores

FMM Iteration Runtime (s)

Parallel Efficiency
Conclusion and Outlook
Conclusion and Outlook

- Octo-Tiger, HPX and the rest of the toolchain work on Ookami
- Node-Level scaling look as expected, distributed scaling needs more (larger) tests
- Kokkos SIMD implementation works using the NEON types
- Kokkos SIMD implementation does not work using the SVE types (yet)

Outlook

- Scaling results with the new Kokkos/HPX implementation
- Get Octo-Tiger to properly use SVE and vectorize more methods
- Benchmark the radiation and port to Kokkos
- Optimize for AMD GPUs

Thanks for your attention! Questions?