Cosmological Particle-Mesh Simulation in Chapel
Ben Albrecht, Nikhil Padmanabhan

Chapel

Chapel is a modern parallel programming language designed for productivity at scale.

Chapel supports:
- General parallelism
- Separation of Parallelism and Locality
- Better suited for emerging heterogeneous architectures
- Multiresolution Design
  - Parallelism and locality interface spans multiple layers of abstraction

Chapel is:
- Open source
  - Licensed under Apache v2, hosted on GitHub
- Portable
  - Runs on laptops, desktops, to Cray supercomputers
- Productive
  - Includes a wide range of modern language features
- A PGAS language
  - Not restricted to static SPM/paradigm

Conclusions and Next Steps

Productivity

Chapel is usable as a productive language today:
- Hybrid parallel/distributed programming is made easy in Chapel, assuming key abstractions are in place.
- It was not difficult to implement the PM code, the FFTW-compatible distribution, and skylight.
- Chapel’s first-class C interoperability feature made interfacing with legacy code simple.
- Interoperability with MPI is functional, though performance is not ideal.
- Tooling was the biggest weakness felt during this work, including:
  - Debugging
  - Profiling
  - Compilation times

Performance

- Chapel performance is within factor of 2 of C+MPI performance
- C code is pure MPI vs. Chapel being hybrid OHEthreads+MPI
- This caused contention when accumulating onto grid in Chapel
- FFTW transposes are single-threaded, which penalizes Chapel’s performance
- Problem size scaling is largely dominated by FFT
- Chapel code uses atomics for assigning particles to grid points, trading performance for simplicity

References
1. N. Padmanabhan, B. Albrecht, "Cosmological Particle-Mesh Simulations in Chapel", Proceedings of the PGAS Application Workshop (P4W), November 2017
2. PM Gitlab: https://gitlab.com/pj/pmm-p4w2017
3. Chapel GitHub: https://github.com/chapel-lang/chapel