Implementations of Particle-Particle and Fast Hierarchical Methods over N-body problem on Heterogeneous platforms

Prasanth Chatarasi, Yuhan Peng, Ankush Mandal

HABANERO Extreme Scale Software Research Group
Department of Computer Science
Rice University

HiPEAC Student Heterogeneous Programming Challenge 2016

November 9, 2016

N-Body Introduction

- Determine the motion of particles over the time due to the forces from the other particles
  - Calculate forces ($\vec{F_i}$) using the potential function
  - Update velocities i.e., $\vec{v_{i+1}} = \vec{v_i} + \Delta t \times \vec{F_i}$
  - Compute new position as $\vec{x_{i+1}} = \vec{x_i} + \Delta t \times \vec{v_i}$

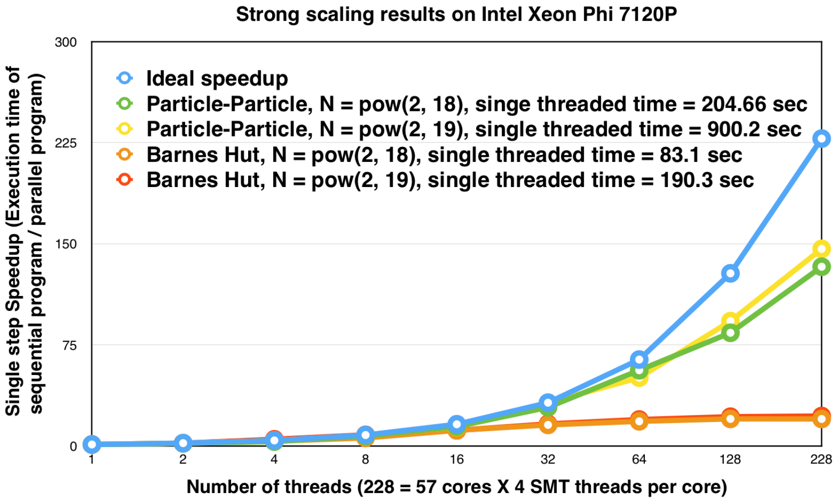
- Various applications in Astronomy, Molecular dynamics, Fluid dynamics etc.

- History of algorithms
  - Exact : Particle-Particle - $O(n^2)$, where n is number of particles
  - Approximation: Mesh based methods - $O(nlogn)$
  - Approximation: Tree based methods - $O(nlogn)$

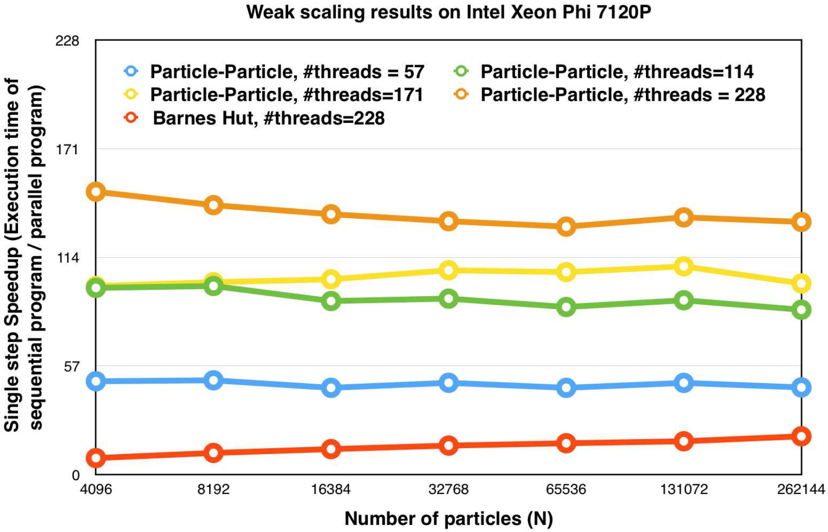
N-Body implementations on Xeon Phi 7120P

- Xeon Phi Co-Processor (Intel Knights Landing) with 57 cores (4 SMT threads per each core) and 512 bit SIMD Instruction set per each core
  - Particle-Particle based approach - Optimization's
    - Array of structures to Structure of Arrays for better vectorization
    - Loop tiling to enhance temporal locality
    - Array expansion to make outermost tile loop parallel
  - Barnes Hut approach (Approximate approach)
    - Idea: When a set of particles are too far away from current particle, we can approximate force to the center of mass of particles rather than calculating for each particle.
    - Enabled OpenMP task-based parallelism to traverse quad tree to compute forces with approximation

Experiments on Intel Xeon Phi for strong scaling - Single step



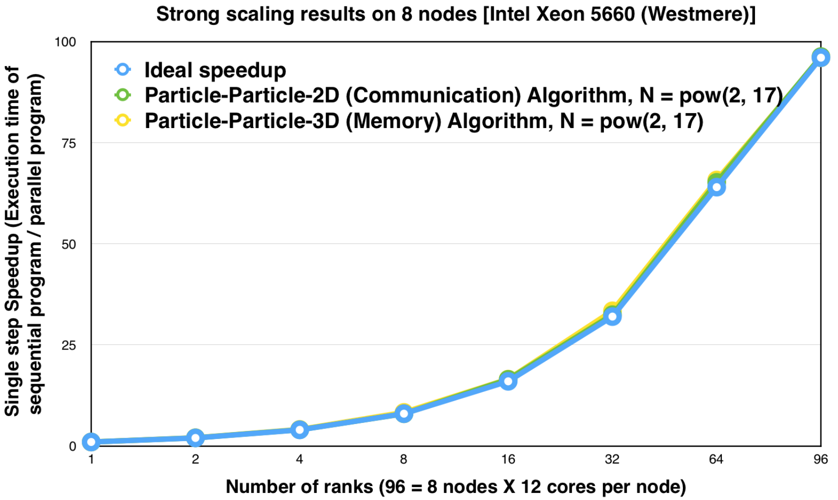
Experiments on Intel Xeon Phi for weak scaling - Single step



Observations on Intel Xeon Phi

- Barnes-Hut's implementation suffers from performance improvement relative to particle-particle based approach
  - Data is distributed across different nodes in the tree, hence resulting in spatial locality loss for vectorization
  - No loop tiling resulting in temporal locality loss
  - Interesting ideas on efficient tree traversals [Milind et.al PLDI 2016]
- Even, particle-particle based approach on Xeon-Phi suffers in case of strong scaling from 57 to 228 threads.
  - Each SMT thread requires use of vector processor unit to perform operations. Huge bottleneck on VPU!
  - Intel Knight's corner announced 2 VPU's per core

Experiments on a cluster (MPI) for strong scaling - Single step



Preliminary experiments with GPU

- Tesla K20 with 13 SMs and 2496 CUDA cores
  - Enabled most of optimization's done in the context of Multi-core
  - Used constant memory for broadcasting small constant values to all the SMs
  - Achieved an occupancy of 61% over $N = 32768$ particles

Learning's and Challenges

- Better acquainted with Intel compilers and Intel Advisor tool for vectorization and multi-threading.
- Fun in working as team in exploring different architectures (Xeon, Xeon-Phi, GPU), algorithms (Particle-Particle, tree based), different programming models (OpenMP, MPI, CUDA), and different parallelism styles (task-based, loop-based, SIMT-based, SPMD-based).

- Hard to efficiently vectorize and optimize for locality in case of tree traversals.