LA-UR-16-22522

A First Look at Performance on the XEON Phi KNL

Timings from a new mini-app: Tycho 2

Kris Garrett

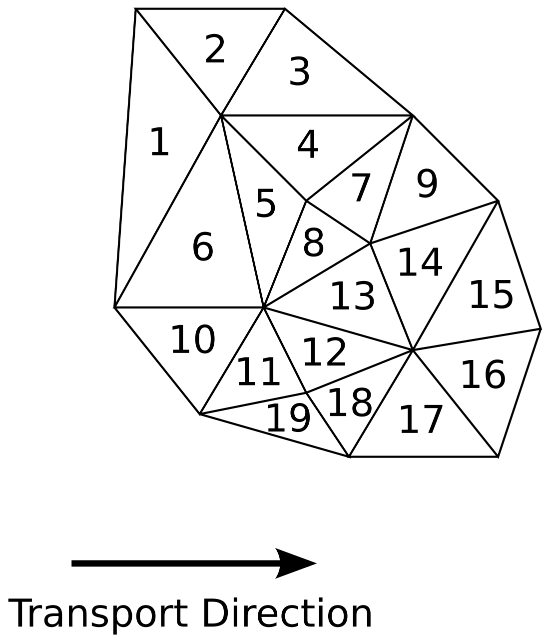
April 19, 2016

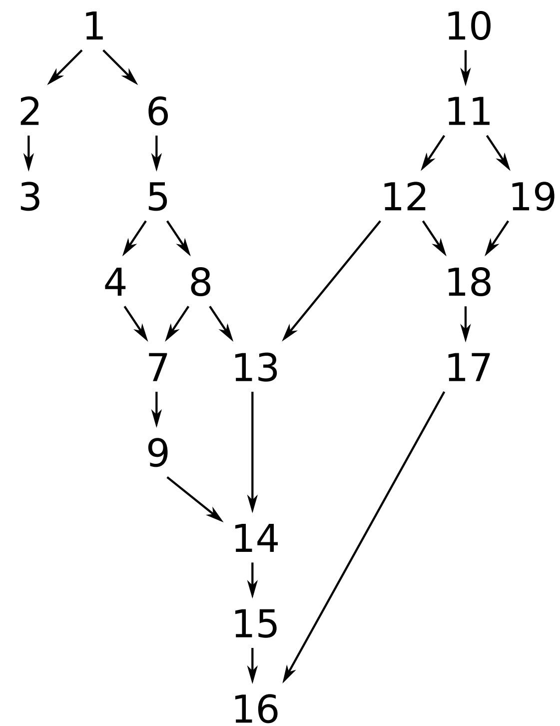
Mini-App: Tycho 2

- **Simulates neutral particle kinetic transport sweeps**
  - Kinetic = function of space and momentum, not just space
- **Unstructured tetrahedral grid**
- **Linear DG in space**
- **Discrete ordinates in angle**
- **Original version created by Shawn Pautz in the early 2000's**
- **New version implements OpenMP**
- **New graph traversal scheduling currently being implemented**
- **Current code has not been heavily optimized, so take timings with a grain of salt**

$$\Omega_q \cdot \nabla_x \Psi_{qg}(x) + \sigma_t \Psi_{qg}(x) = Q_{qg}(x)$$

Mini-App: Tycho 2



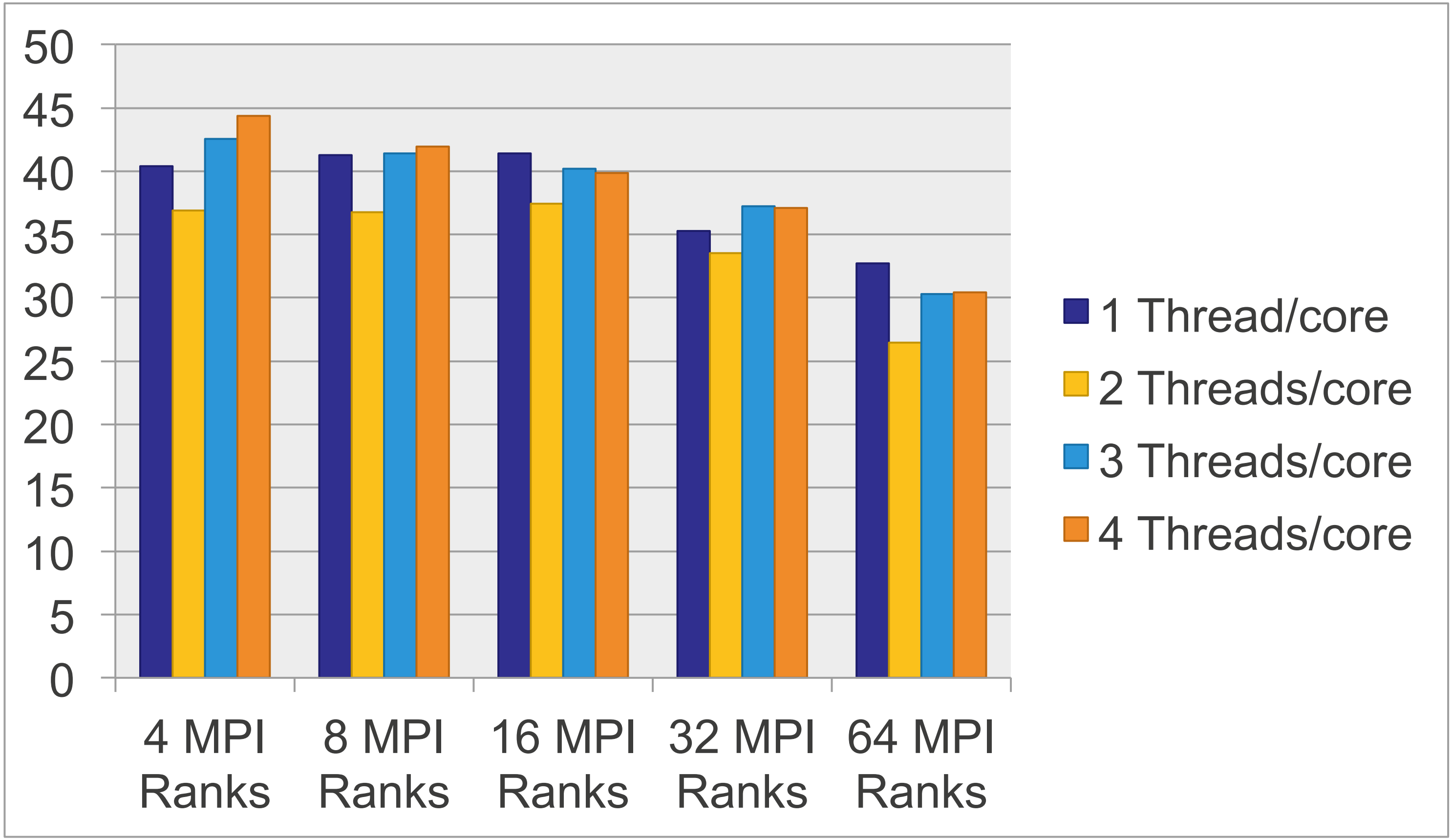


$$\Omega_q \cdot \nabla_x \Psi_{qg}(x) + \sigma_t \Psi_{qg}(x) = Q_{qg}(x)$$

Performance on B0 KNL

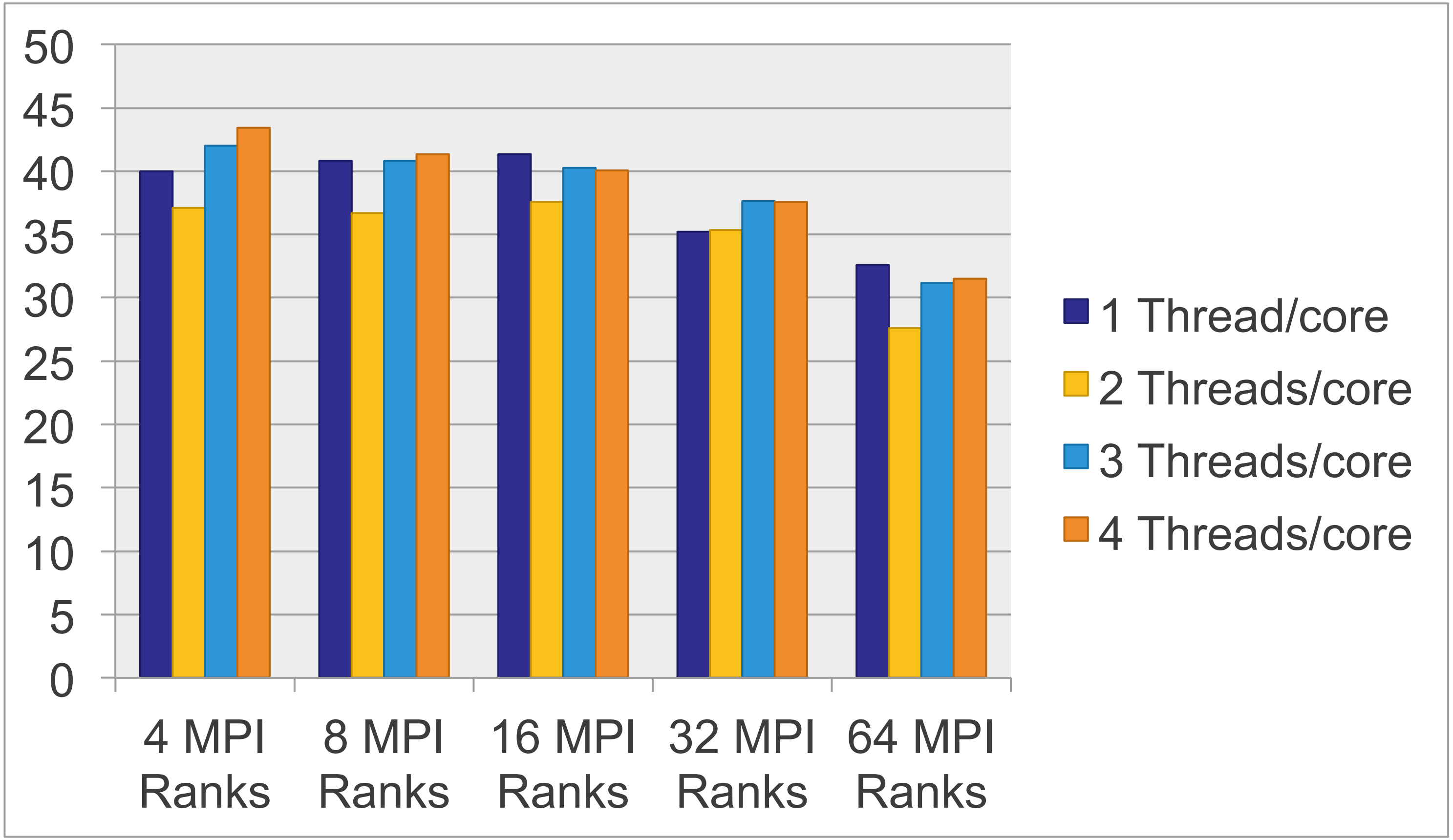
- **Problem setup**
  - Approximately 10,000 cells
  - 200 angles (q)
  - 10 groups (g)
- **Hardware**
  - 1 KNL with 64 cores
  - Each core can switch between up to 4 hardware threads (1,2,3,4)
  - Fast 16GB MCDRAM that can be used as an explicit/implicit cache
  - 2 vector processing units per core

Performance on B0 KNL: Cached MCDRAM



*Threads fill up cores (ex. 4 MPI Ranks, 1 Thread/core implies 16 threads per MPI Rank)

Performance on B0 KNL: Non-Cached MCDRAM



*Threads fill up cores (ex. 4 MPI Ranks, 1 Thread/core implies 16 threads per MPI Rank)

Performance on B0 KNL: Takeaways

- **No special code needed to compile/run on KNL**
- **Best single node runs: very few threads, many MPI tasks**
- **Even all MPI works well for this application**
  - 128 MPI ranks and no threading: 28.07s
  - 64 MPI ranks and 2 threads: 26.45s
- **No-cache vs cache mode yields roughly the same performance**
  - Cache 64 MPI ranks and 2 threads: 26.45s
  - No Cache 64 MPI ranks and 2 threads: 27.62s
  - ***Warning***: this code has not been optimized for memory accesses yet which is probably why the cache has very little effect

My Thoughts on Performance Portability for KNL

- **KNL has approximately twice as many cores at half the processor speed**
  - With no special programming, KNL should be competitive with current CPUs for most codes
  - Only true IF all cores are used for most of the code
    - Another case for many MPI ranks and few threads
    - Or SPMD threading paradigm
    - Setup code needs to utilize most/all cores, everything must be parallel
- **Each core has 2 vector processing units**
  - Oversubscribing cores by at least 2 is probably best

My Thoughts on Performance Portability for KNL

- **Highly vectorized code**
  - Useful for all architectures
  - KNL has wider vector lengths than other CPUs, so this will help KNL more
- **Accelerator code requires explicitly moving data to/from device**
  - Maybe the same area of the code can be used to explicitly cache data into MCDRAM for the KNL
- **Use tiling of large data structures and make tile sizes a compiling parameter or runtime parameter**
  - Can create tiles to easily fit into MCDRAM for caching
  - Useful for moving data to/from accelerators
- **Overall: good CPU performance = good KNL performance**

The End


CRAY
Alliance for Computing at Extreme Scale
NNSA
Los Alamos
TRINITY