THE UNIVERSITY OF CHICAGO

Effective OpenMP Implementations

Yuliana Zamora

THE UNIVERSITY OF NEW MEXICO — EST.1889

Los Alamos NATIONAL LABORATORY — EST.1943

Abstract

OpenMP is a programming model to increase on node parallelism in applications. It often comes under attack for poor performance compared to MPI everywhere. This is because OpenMP is conventionally used at the loop level and thus suffers from high thread start up costs and thread synchronizations. A higher-level implementation of OpenMP can reduce the typical overhead by having the parallel region encompass the whole main loop and partitioning the child loops statically. We apply this method within a variety of software applications (SELF, HIGRAD, CLAMR) to investigate speed-ups due to reductions in OpenMP overhead, thread starts, and thread waiting times by the implementation of high-level OpenMP.
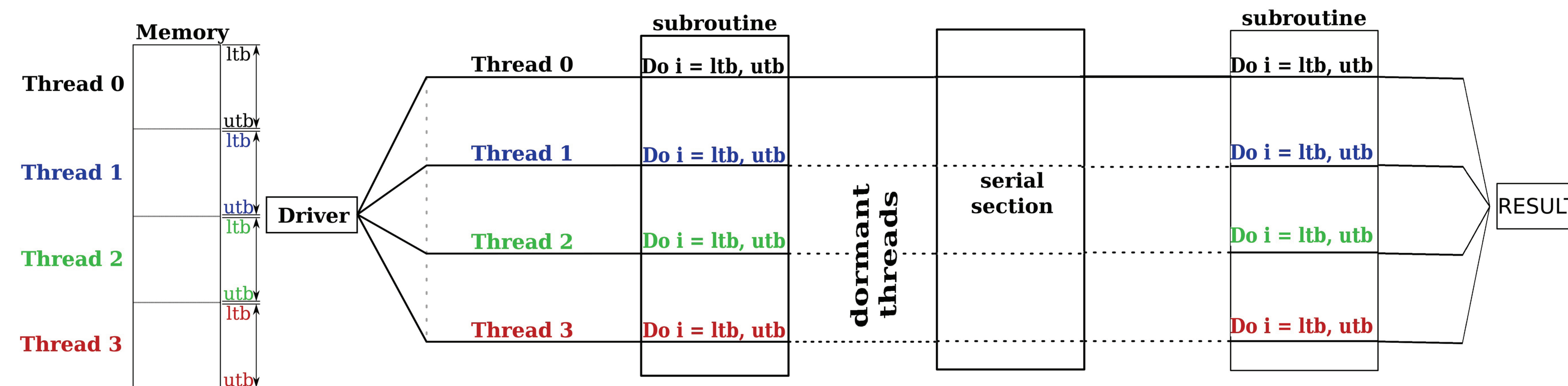
High Level OpenMP

Figure 4. Implementation of high-level OpenMP using 4 threads going through two subroutines. Figure shows threads are 'alive' through both parallel region and serial regions of the code.
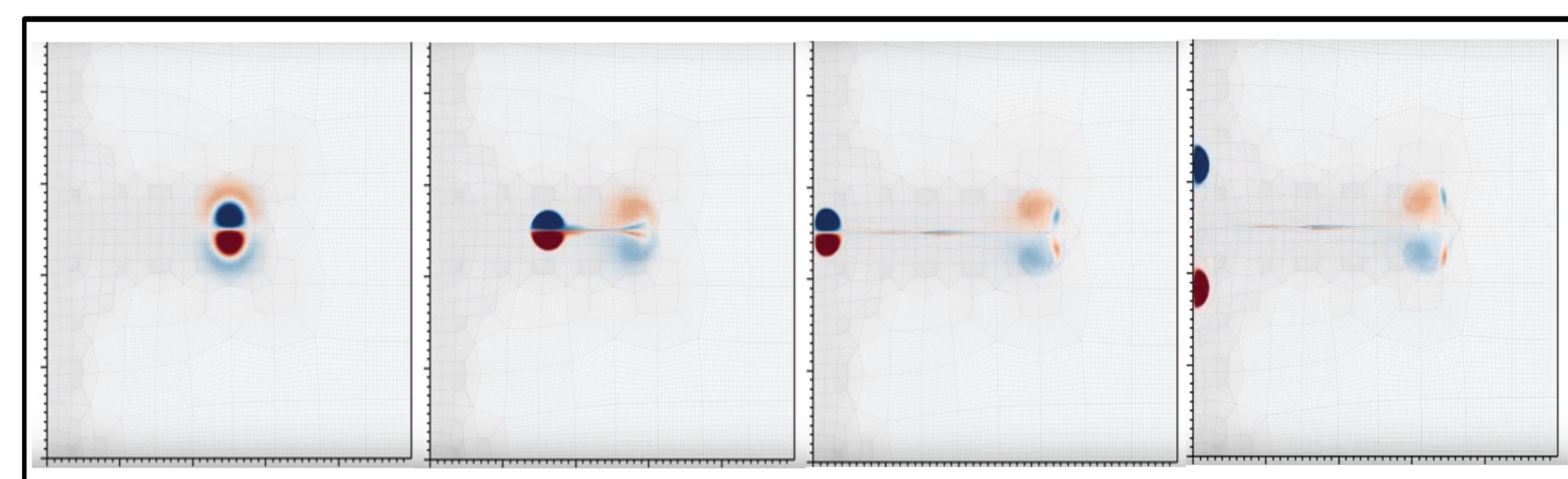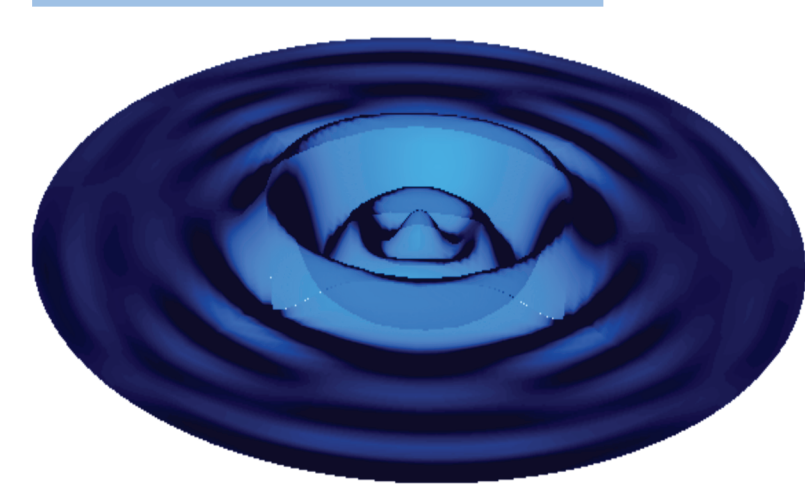
Applications

SELF

Figure 1.
The Spectral Element Libraries in Fortran (SELF) is an open-source library that houses routines necessary to implement spectral element methods. The shallow-water solver (Dipole example) is a demonstration of the SELF applied to 2-D hyperbolic conservation law on an unstructured mesh.

HIGRAD

Figure 2. An atmospheric hydrodynamics model, HIGRAD, is coupled to a wildfire behavior model, FIRETEC, to produce a coupled atmosphere/wildfire behavior model based on conservation of mass, momentum, species, and energy. HIGRAD/FIRETEC is a three-dimensional transport model that uses a compressible-gas formulation to simulation the coupling between wild land fire and motions of the local atmosphere. Figure and summary taken from reference [1].
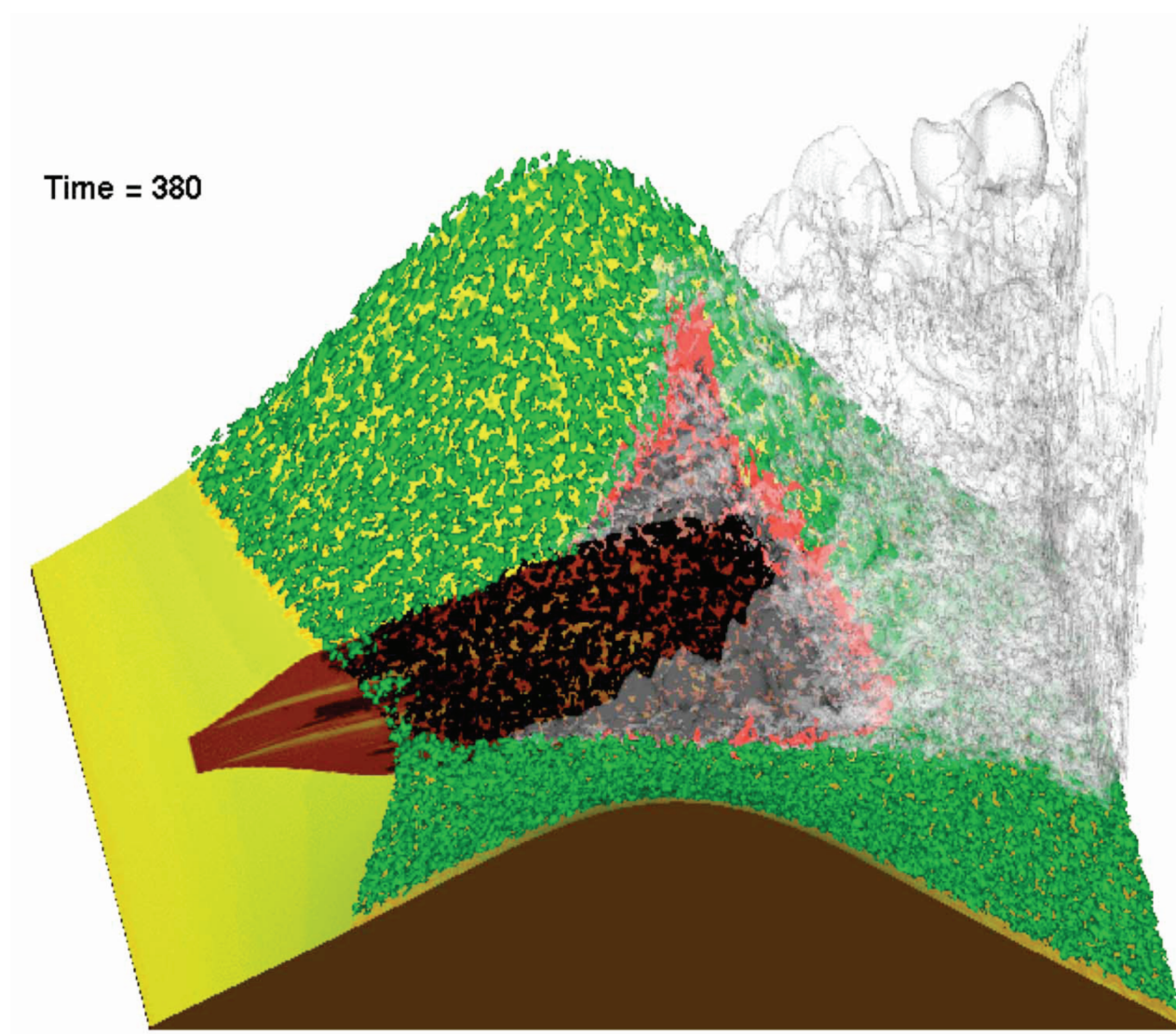
CLAMR

Figure 3. CLAMR (Compute Language Adaptive Mesh Refinement) is being developed as a DOE mini-app. CLAMR is being used to develop the computer science infrastructure needed for cell-based adaptive mesh refinement to effectively run on an Exascale class system. As a simple representative physics model, the shallow water equations are used. This kind of physics model can be used to simulate tsunamis and many other water flows.[3] Figure taken from reference [2].
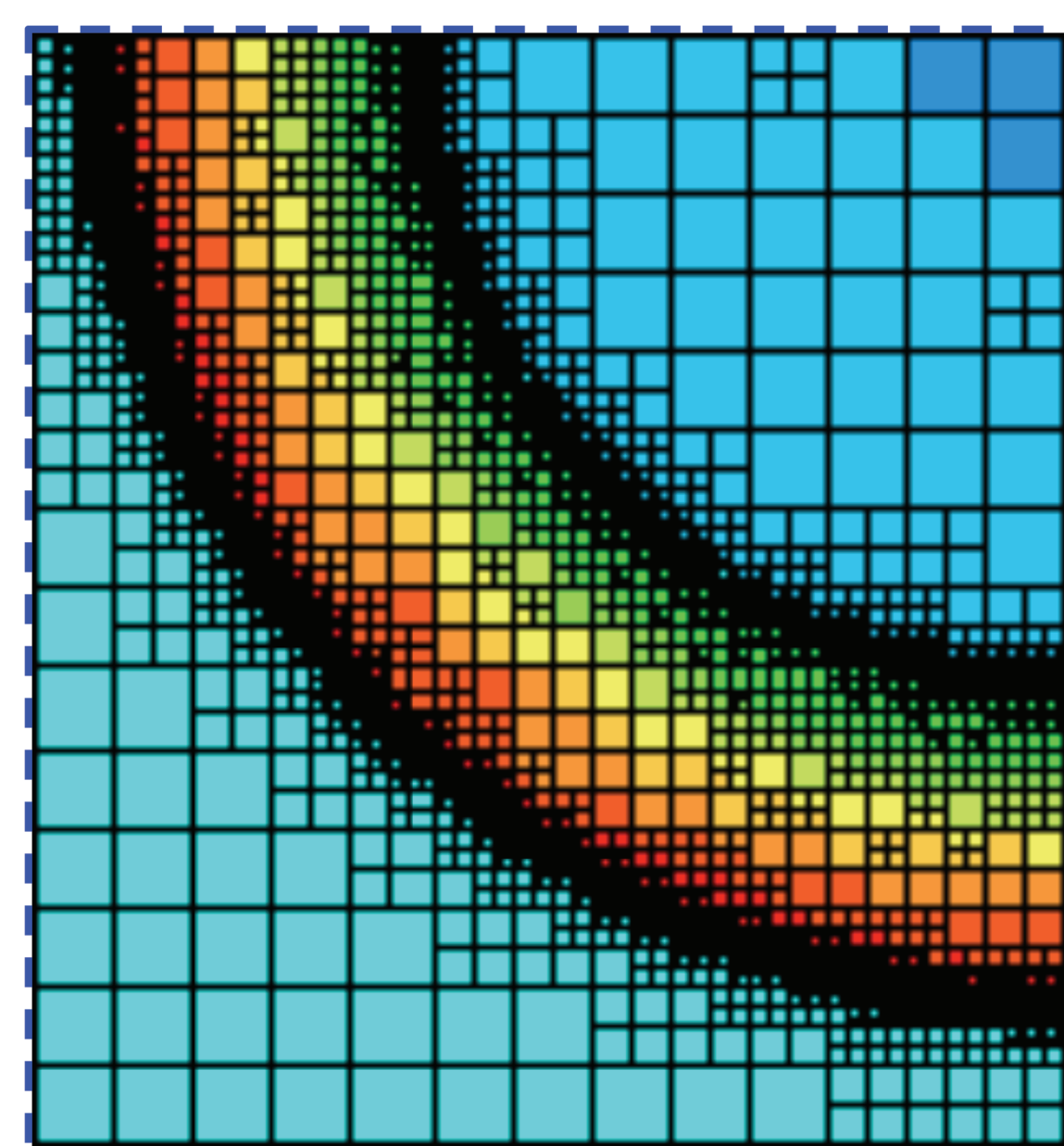
Results

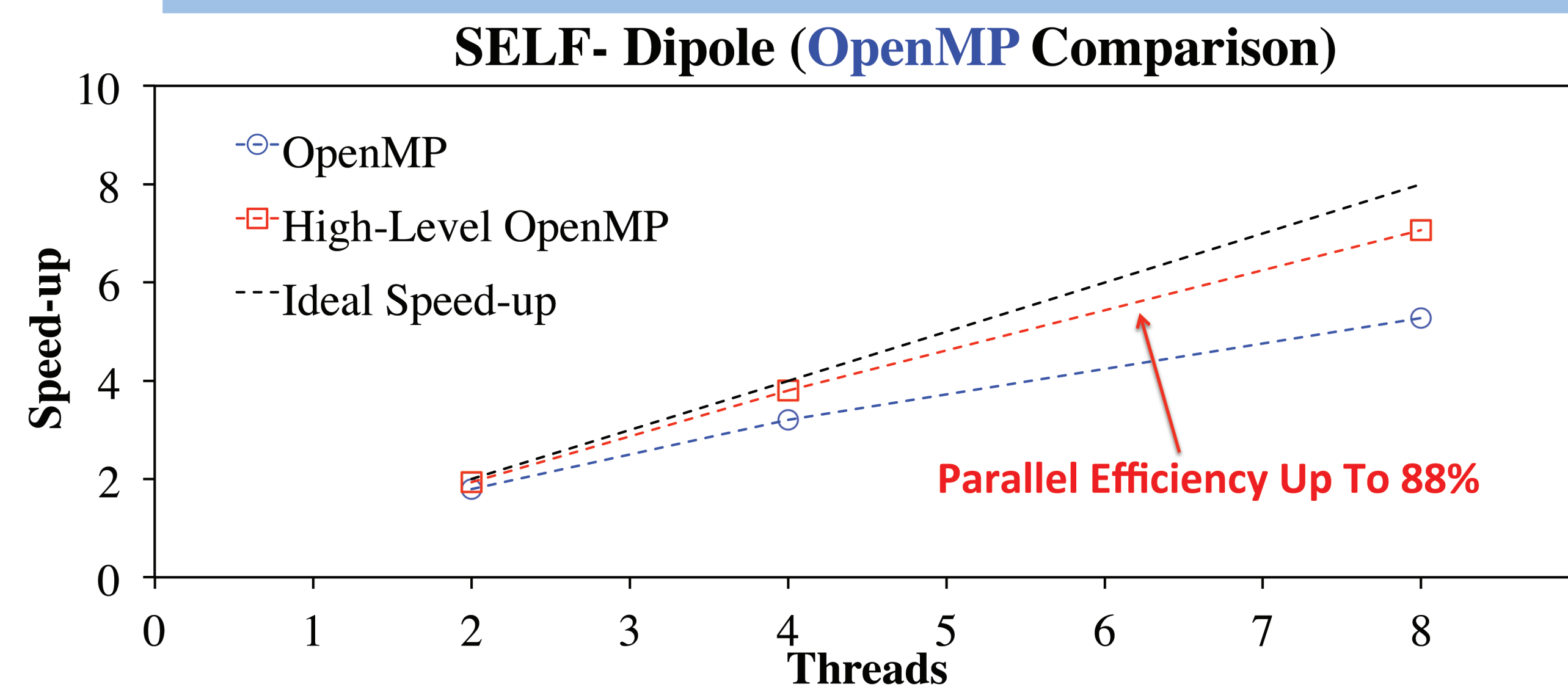### Eight Core Intel Xeon SandyBridge E5-2670

Figure 5. Plot of high-level OpenMP speed-up compared to ideal speed-up and conventional OpenMP implementation.

36 Core Intel Broadwell E5-2695 v4

Figure 6. Plot of high-level OpenMP speed-up compared to MPI only and MPI+OpenMP.

Figure 7. Plot of high-level OpenMP speed-up compared to MPI only. There is more work needed on CLAMR for a full implementation of HLOMP. Tests run on a Broadwell E5-2600 36 core processor.

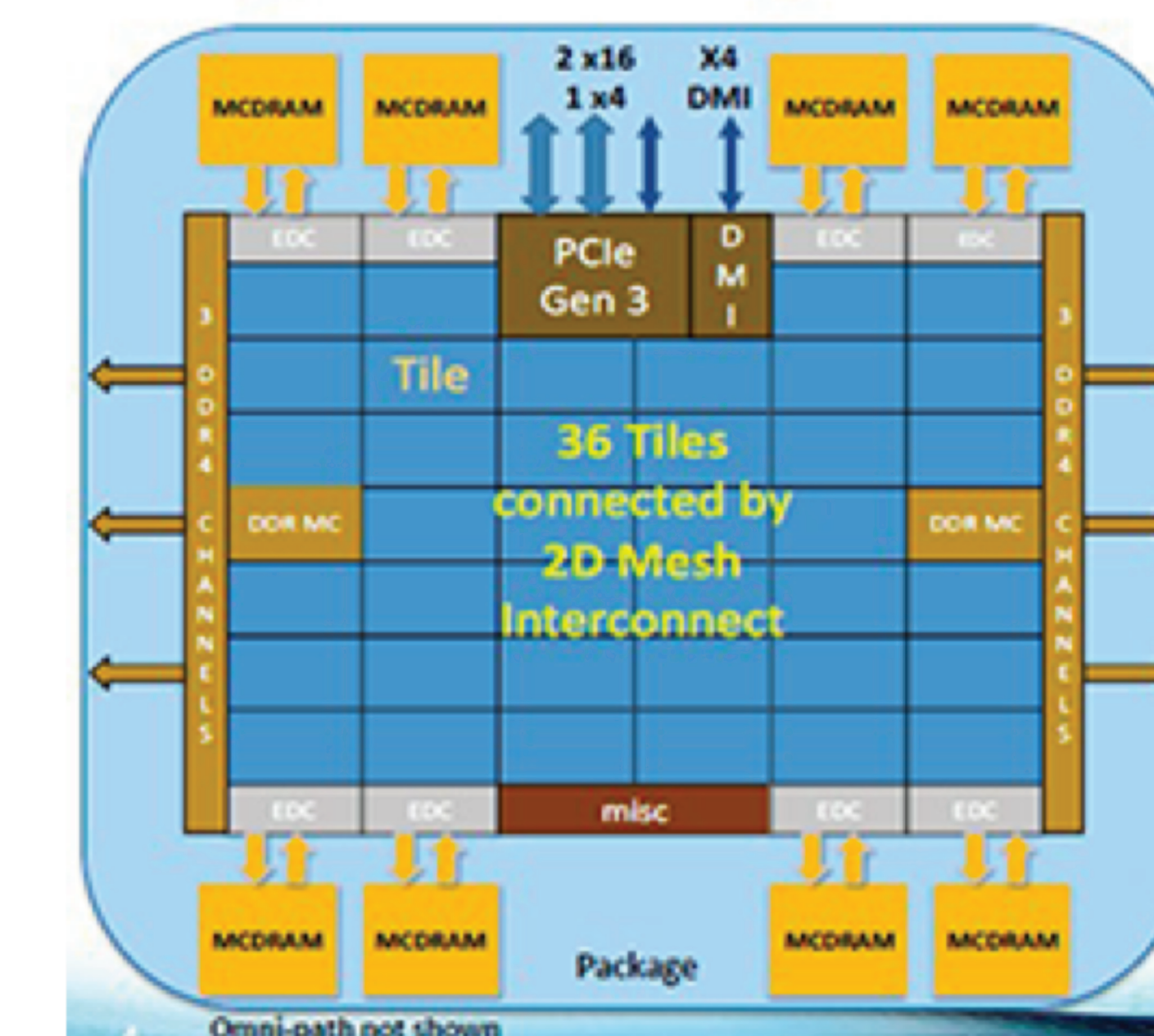### Intel Xeon Phi 7230 Processor (Knights Landing)

Figure 8. Intel's Knights landing architecture is a many core processor. The configuration we are using has 64 cores with 4 threads each. Figure taken from[4].
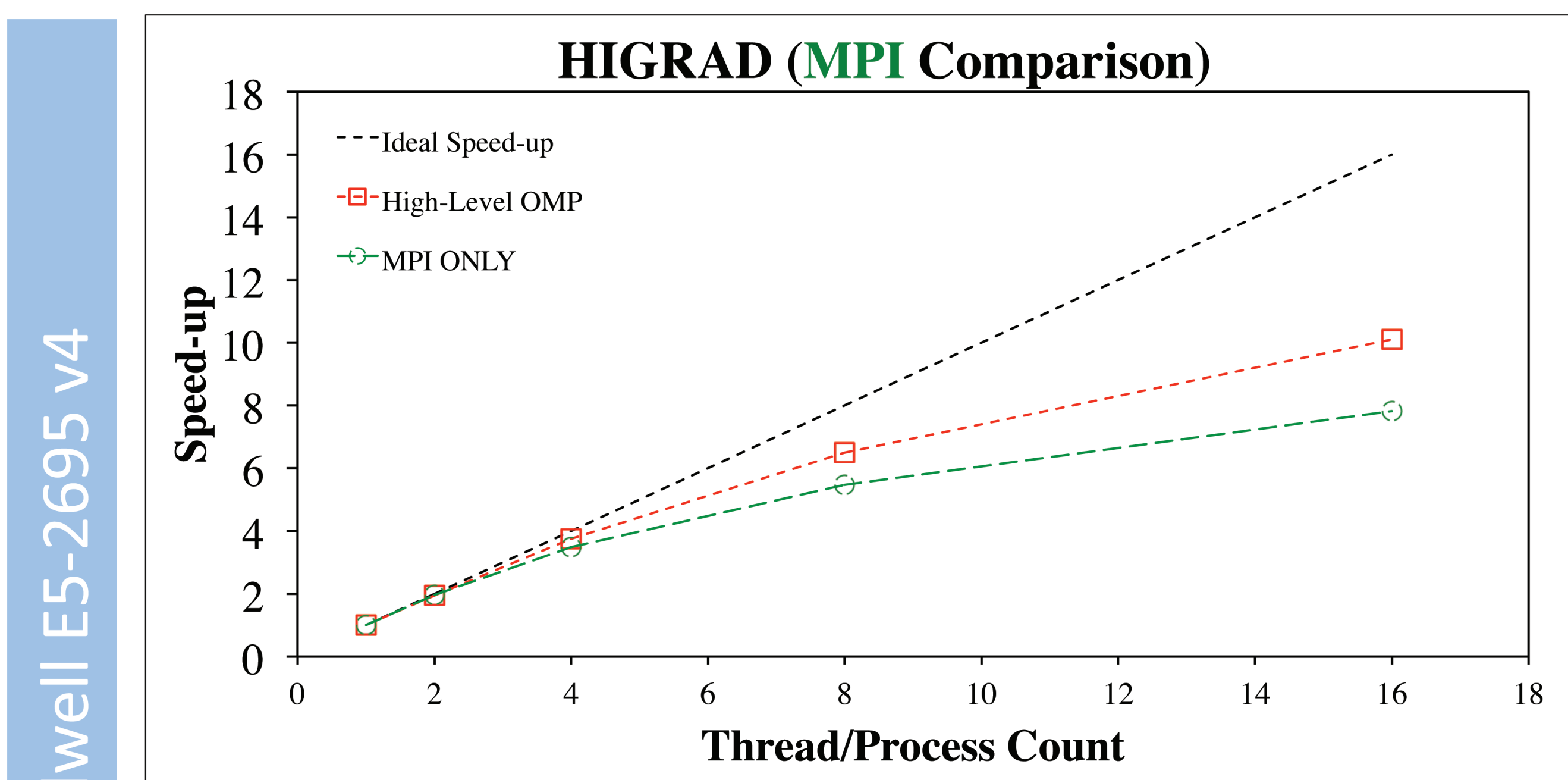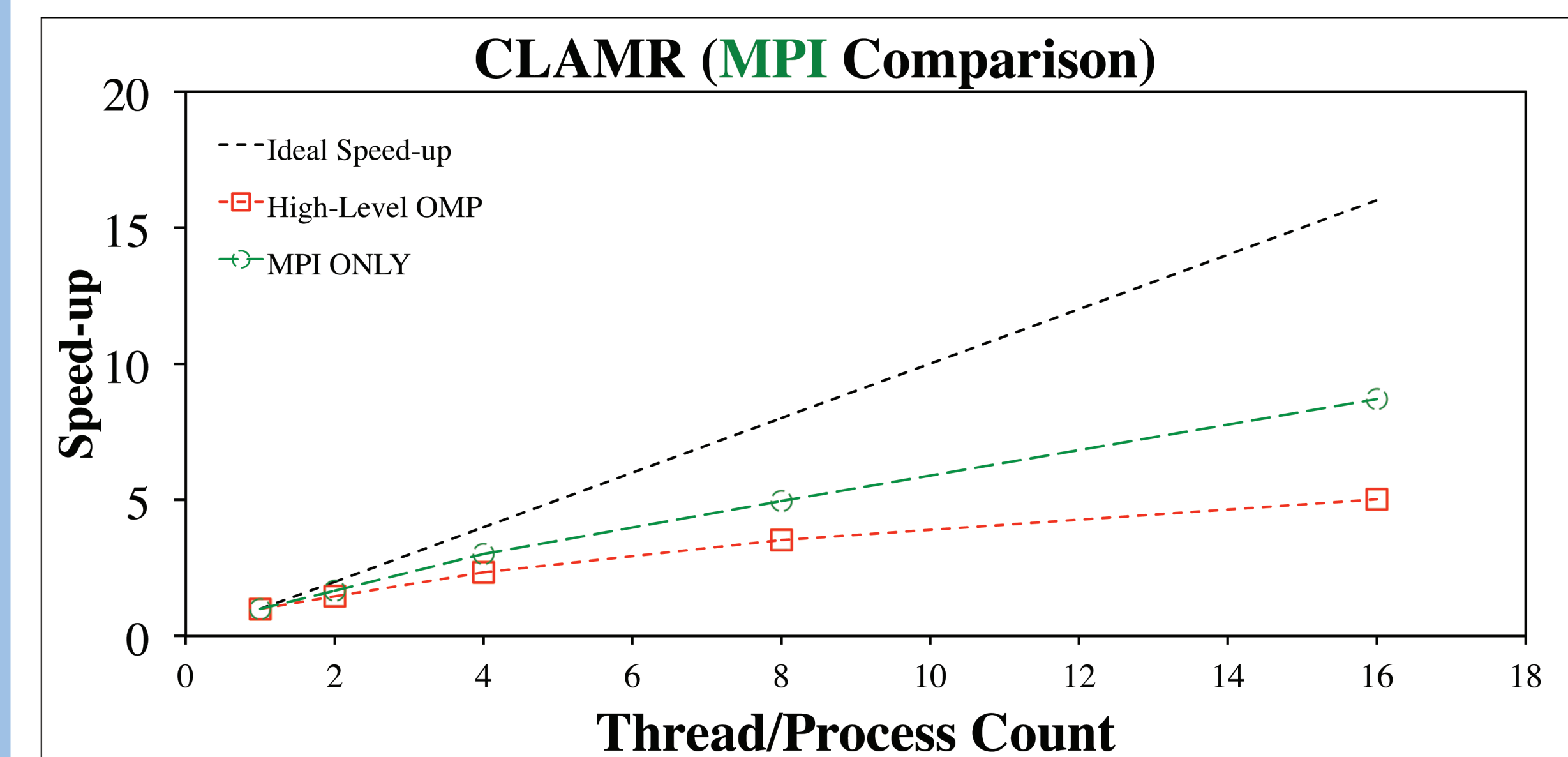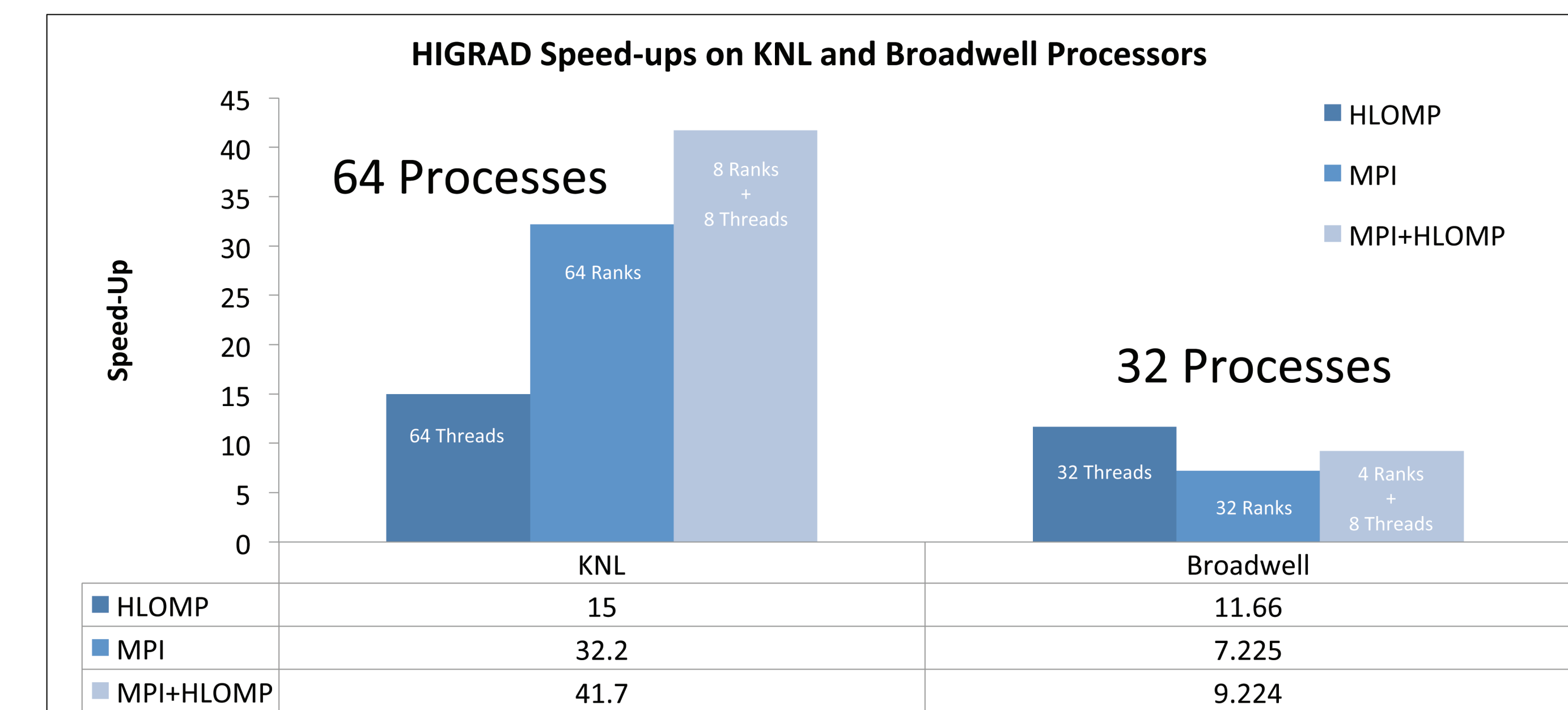

	KNL	Broadwell
HLOMP	15	11.66
MPI	32.2	7.225
MPI+HLOMP	41.7	9.224

Figure 9. Plot of Higrad on a Quad-Cache Mode configuration of a KNL node and Broadwell processor. The improvement in the high-level OpenMP on Broadwell is due to a full implementation of HLOMP. Using up to 90 nodes, with a mesh of 10,000 x 10,000 x 170, a simulation of 17 billion cells was able to efficiently run on a KNL processors.

Conclusion & Future Work

High-level OpenMP is applied to three different software applications, with different hardware. The speed-ups obtained by introducing high-level OpenMP compared to the conventional approach in SELF is significant (up to 88% parallel efficiency). The implementation of high-level OpenMP in HIGRAD and CLAMR demonstrates comparable speed-ups to MPI (OpenMPI). The ability to use the same thread address across child classes allows for these great speed-ups with minimal fundamental changes in the original code. The current results on the KNL reveal that further testing and configurations need to be implemented to see better speed-ups and equivalent results.

References:
[1] http://www.lanl.gov/orgs/ees/ees16/FIRETEC.shtml
[2] http://www.lanl.gov/projects/feynman-center/technologies/software/clamr.php
[3] Rebecca Tumblin, Peter Ahrens, Sara Hartse, Robert W. Robey *Parallel Compact Hash Algorithms for Computational Meshes* SIAM Journal of Scientific Computing (Feb. 2015)
[4] http://www.nextplatform.com/2016/06/20/intel-knights-landing-yields-big-bang-buck-jump/ nkelman G, Uberuaga BP, Jonsson H. *J. Chem. Phys.* 113:9901-4 (2000)

*Special thanks to Bob Robey, Joe Schoonover, and Dave Nystrom