Optimizing Molecular Dynamics and Stencil mini-applications for Intel's Knights Landing

Kavitha Chandrasekar, Laxmikant V. Kale
University of Illinois Urbana Champaign

I. Abstract

To achieve the best performance and energy efficiency, HPC applications may require tuning on new architectures like Knights Landing, Intel's 2nd generation Xeon Phi processor. Furthermore, different applications with varying characteristics might benefit from variant configurations and tuning.

In order to study the different types of applications, we select a molecular dynamics kernel, LeanMD [1] and Stencil3D code from the Charm++ [2] benchmark suite, which are representative of compute and communication intensive HPC benchmarks, respectively.

We analyze these applications on different KNL configurations, namely MCDRAM usage mode for a given cluster mode and perform within node optimizations, specfically use of hyperthreading and enabling CPU affinity.

With this tuning of applications based on their characteristics, we show a performance improvement of upto 1.8X when enabling hyperthreading and energy savings of nearly 10%.

II. Applications

LeanMD is a molecular dynamics application which computes the forces on atoms in cells and based on the calculated forces, the atoms velocity and positions in the cells are determined. LeanMD is computationally intensive and exhibits high load imbalance. We use GreedyLB load balancer to perform load balancing in our experiments.

Stencil3D performs high data exchange between its neighboring cells in a 3 dimensional space. It is a communication intensive application and is a representative of stencil codes like iterative Jacobi solvers. We use GreedyLB load balancer to ensure load is balanced in our experiments.

We use the Charm++ framework to run these applications. The experimental runs have been made on a single node. We plan to extend our studies on the Stampede 2.0 KNL cluster in the future.

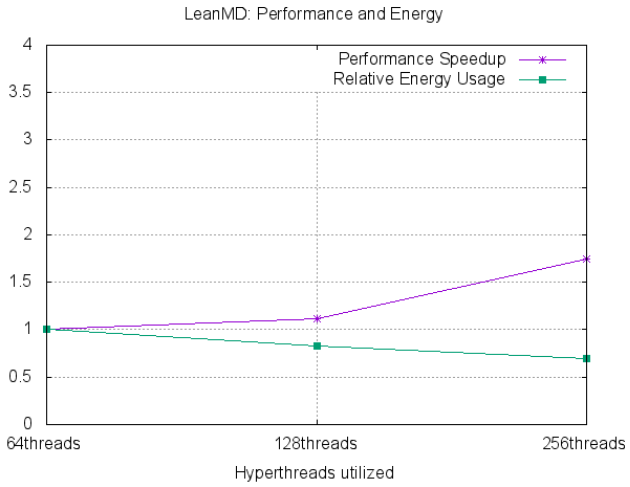

Fig. 1: Performance of LeanMD with Hyperthreading.

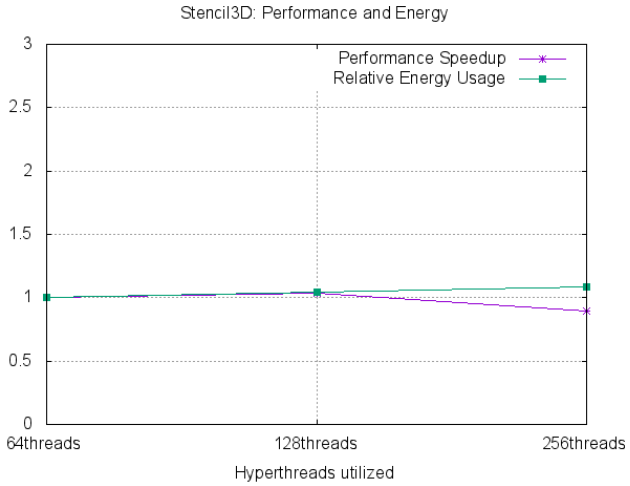

Fig. 2: Performance of Stencil3D with Hyperthreading.

III. Experimental Results

Experiments were on a single KNL node with MCDRAM in Cache mode. For MCDRAM configuration comparison, both Flat mode and Cache mode configurations were used. The clustering mode used was Quadrant mode.

For LeanMD, it was observed that with hyperthreading, the application benefitted the most when using all 4 hyperthreads as seen in 1. The graph shows energy usage (lower the better) and speedup (higher the better). We measure performance after load balancing phase, since load balancing is a one-time, but varying overhead in each case.

For Stencil3D, the peformance degraded by about 10% when using four hyperthreads per core and energy usage was close to 10% higher when using 4 hyperthreads per core. Best energy usage and within 5%of best performance was obtained when utilizing one hyperthread per core, as seen in 2. The amount of memory accesses is seen in 3.

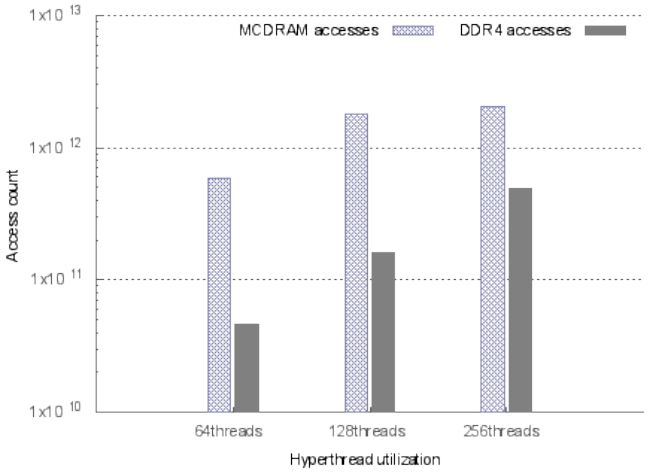

Fig. 3: Amount of memory accesses by Stencil3D with Hyperthreading.

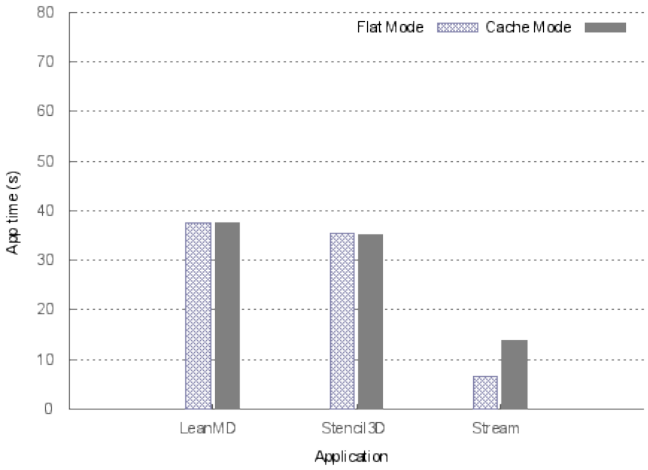

Fig. 4: Performance comparison of applications using MCDRAM in Flat vs Cache Mode.

CPU-pinning We saw close to 30% performance degradation without CPU-pinning. This can be attributed to performance loss from lack of L2 cache locality and poor load balancing when migrating workers across processors with varying frequencies.

We compare the performance of applications when using MCDRAM in Cache vs Flat mode, when the working set size fits within the MCDRAM, in 4.

Stream is the only application which suffered from performance degradation when running on cache mode. The potential for reduced miss penalty with prefetch and avoiding thrashing due to conflicts between near-simultaneous addresses as seen in Stream is where Flat mode is expected to perform better than Cache mode.

IV. Conclusion

We have demonstrated that HPC applications like LeanMD and Stencil3D can benefit in terms of performance and energy when tuned appropriately on the 2nd generation Xeon Phi processor, Knights Landing. We plan to automate this tuning for application characteristics within the Charm++ framework by adjusting architectural knobs like hyperthreading, cpu affinity and high-bandwidth memory usage modes.

References

[1] "leanmd," http://charm.cs.illinois.edu/research/leanmd.

[2] L. Kalé and S. Krishnan, "CHARM++: A Portable Concurrent Object Oriented System Based on C++," in *Proceedings of OOPSLA'93*, A. Paepcke, Ed. ACM Press, September 1993, pp. 91–108.