2018 IEEE International Conference on Data Mining Workshops (ICDMW)

Parallel k-means Clustering of Geospatial Data Sets Using Manycore CPU Architectures

1st Richard Tran Mills
Mathematics and Computer Science Division
Argonne National Laboratory
Lemont, IL, USA
rtmills@anl.gov

2nd Vamsi Sripathi
Data Center Group
Intel Corporation
Hillsboro, OR, USA
vamsi.sripathi@intel.com

3rd Jitendra Kumar
Environmental Sciences Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA
jkumar@climatemodeling.org

4th Sarat Sreepathi
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA
sarat@ornl.gov

5th Forrest M. Hoffman
Computational Science and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA
forrest@climatemodeling.org

6th William W. Hargrove
Southern Research Station
USDA Forest Service
Asheville, NC, USA
hnw@geobabble.org

***Abstract*—The increasing availability of high-resolution geospatiotemporal data sets from sources such as observatory networks, remote sensing platforms, and computational Earth system models has opened new possibilities for knowledge discovery and mining of weather, climate, ecological, and other geoscientific data sets fused from disparate sources. Many of the standard tools used on individual workstations are impractical for the analysis and synthesis of data sets of this size; however, new algorithmic approaches that can effectively utilize the complex memory hierarchies and the extremely high levels of parallelism available in state-of-the-art high-performance computing platforms can enable such analysis. Here, we describe *pKluster*, an open-source tool we have developed for accelerated k-means clustering of geospatial and geospatiotemporal data, and discuss algorithmic modifications and code optimizations we have made to enable it to effectively use parallel machines based on novel CPU architectures—such as the Intel Knights Landing Xeon Phi and Skylake Xeon processors—with many cores and hardware threads, and employing significant single instruction, multiple data (SIMD) parallelism. We outline some applications of the code in ecology and climate science contexts and present a detailed discussion of the performance of the code for one such application, LiDAR-derived vertical vegetation structure classification.**

I. Introduction

Observational and modeled Earth science data span vast temporal and spatial scales. Due to advances in sensor development, dramatic increases in computational capacity, and growing data storage densities, the volume, complexity, and resolution of Earth science data are rapidly increasing. While these heterogeneous, multi-disciplinary, geospatial data offer new opportunities for scientific discovery, the resulting explosion of data has rendered traditional analysis methods ineffective. The promise of scientific advances from this wealth of data has stimulated development and application of data mining, machine learning, and information theoretic approaches—often on large parallel supercomputers—for combining, integrating, and synthesizing Earth science data. Cluster analysis [1] has proven useful for segmentation, feature extraction, change detection, and network analysis with a variety of geospatial and geospatiotemporal Earth science data. Specifically, cluster analysis is used for delineating ecoregions [2], [3], stratifying climate regimes [4], quantifying the representativeness of sampling networks [5]–[8], and classifying vertical vegetation structure from airborne LiDAR data [9]—the use case we work with in this paper.

As the volume of available Earth science data has grown, so too has the power of the computational platforms available for their analysis. In the past decade, however, much of this growth in computational power has relied on increasing amounts of parallelism, in the form of GPGPUs and multi- and many-core CPUs that employ vector processing units. Traditional software tools used in geostatistics, designed for sequential processors, are unable to harness the computational power of these resources and are therefore unsuitable for analysis of these growing Earth science data sets, especially on new emerging architectures. In this study, we explore strategies to enable scalable, parallel performance of k-means clustering—one of the most widely-used tools for unsupervised classification in geostatistical applications—targeting the second-generation Knights Landing Intel Xeon Phi, as well as current and recent generation Intel Xeon server processors with large core counts and reliance on AVX2 and AVX-512 vector instruction sets.

II. A distributed k-means clustering code for geospatio(temporal) applications

A. Basic k-means clustering implementation

We work with a distributed memory parallel k-means code—which we have recently renamed *pKluster*—with a long history. *pKluster* was originally developed in the mid 1990s for clustering large ecological data sets on early "Beowulf"-style distributed-memory cluster computers constructed out

of surplus parts [10]. Because of the extreme heterogeneity of the clusters, a master-worker parallel programming paradigm (implemented using the then new Message Passing Interface, MPI) was used [3], [11], as this provided excellent dynamic load-balancing. On modern, homogeneous machines, the master-worker paradigm may be less efficient than a fully distributed, masterless approach; and thus we developed a scalable masterless k-means clustering code [12]. However, some of the techniques described below introduce load imbalance even on homogeneous machines. Here, we work with the master-worker version of the code. When *pKluster* was initially written, on-node parallelism was virtually nonexistent on commodity PCs (a few had superscalar processors that offered some instruction-level parallelism); the focus was purely on distributed-memory parallelism.

The k-means algorithm is widely used to classify members of a data set consisting of n observation vectors $(x_1, x_2, \ldots, x_n)$, each of dimension m, into k clusters, based on some measure of similarity (we use Euclidean distance here, but other metrics, such as cosine similarity, are possible as well). The number of clusters, k, is a prescribed, fixed parameter. This iterative algorithm starts with a collection of k "seed centroids" $(z_1, z_2, \ldots, z_k)$, and computes the distance of each observation vector x to each centroid; the observation is then assigned membership in the cluster associated with the closest centroid. After all observations have been classified, a new set of centroids is calculated by computing the centroid (vector mean of all assigned observations) for each cluster. Iteration continues until only a small proportion (we use $< 0.05\%$) of observations change their cluster assignment. Our implementation of a master-worker parallel version of this algorithm is straightforward. The master process assigns an *aliquot* of observations to each worker process, and, for each observation, the workers compute the distance to each centroid, assign the observations to the closest cluster, and report the new cluster memberships to the master. As aliquots are completed, the master assigns additional aliquots to workers from its task queue. When all aliquots have been processed, the iteration ends and the master computes the new centroids and broadcasts them to the workers. The aliquot size is a tunable parameter. Small aliquots enable better load balance, but increase the number and volume of MPI messages and the demands placed on the master process. In this study, we divide the data set into a number of aliquots equal to the number of worker MPI processes. This allows each worker process to work with the same set of observations throughout the calculation, obviating the need for great deal of I/O, but at the expense of losing any load balancing inherent in the master-worker model. Prior experience on modern homogeneous clusters has indicated that this is generally a reasonable choice when running across multiple nodes.

B. Triangle inequality-based acceleration

For very large datasets or cases when the number of clusters k is large, straightforward implementation of k-means proves too expensive, despite the ability to harness many compute nodes. *pKluster* "accelerates" the k-means process by using two techniques described by [13], [14]. First, the triangle inequality is employed to eliminate unnecessary point-to-centroid distance computations based on the previous cluster assignments and the new intercentroid distances; this approach is illustrated schematically in Figure 1. The expense of distance calculations is further reduced by maintaining a sorted list of the intercentroid distances: when searching for the closest centroid to an observation p, the new candidate centroids c_j are evaluated in order of their distance from the former cluster centroid c_i. Once the critical distance $2d(p, c_i)$ is surpassed, no additional evaluations are needed, since the nearest centroid is known from a previous evaluation. Figure 2 illustrates the dramatic reduction in time spent in distance calculations that the acceleration technique enables. Typically, the number of distance comparisons per iteration rapidly drops off as the k-means clustering progresses—in fact, we have yet to encounter a dataset for which this behavior is not observed.

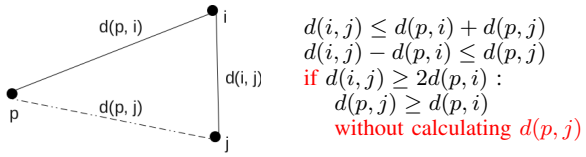


Fig. 1: Illustration of how the triangle inequality is used to eliminate unnecessary distance calculations.

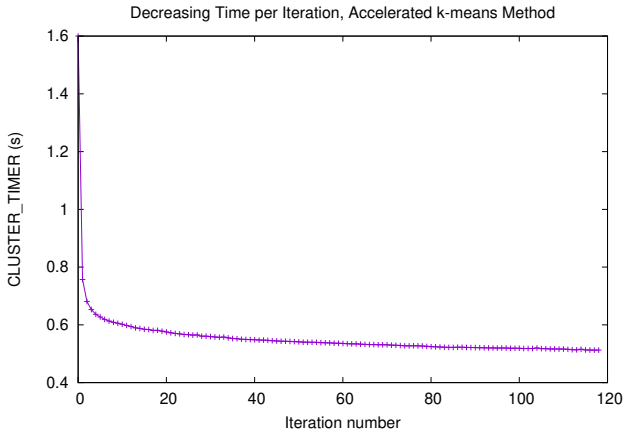


Fig. 2: Timings per iteration for clustering the GSMNP LiDAR dataset for $k = 2000$ on a dual-socket Intel ("Broadwell-EP") Intel® Xeon® E5-2697 v4 machine (18 cores per socket) using the accelerated k-means algorithm. The time required for each iteration follows a decreasing trend because the accelerated algorithm is able to avoid many distance comparisons.

III. Manycore CPUs and the Intel Xeon Phi

Although transistor density has continued to increase, "power wall" considerations have limited practical CPU frequencies to around 4 GHz since the middle of the 2000s, when feature size reached 65 nm, below which Dennard

(or MOSFET) scaling no longer holds [15]. To continue performance gains from one microprocessor generation to another, CPU manufacturers now largely rely on increasing levels of on-chip parallelism. In recent years, the number of CPU cores and hardware threads has dramatically increased: "manycore" processors, such as the Intel Xeon Phi line have come to market, and this trend is also observed in standard server processors. In addition to increases in core and thread counts, there is increasing reliance on fine-grained parallelism in the form of data-parallel vector processing units (VPUs) that operate on many SIMD (single instruction, multiple data) lanes. Recent generations of Intel Xeon processors have 256-bit vector registers and support the AVX2 instruction set and fused multiply-add (FMA) instructions, and the current "Skylake" generation of Xeon processor has 512-bit vector registers and supports the AVX-512 instructions.

The trend towards increased parallelism is particularly exemplified in the second-generation Intel Knights Landing (KNL) CPU [16], a manycore processor with up to 72 compute cores, each of which has FMA-capable VPUs operating on 512-bit vector registers. Because exploring approaches to obtaining good performance out of this architecture is the main motivation in this paper (though the code optimizations we have implemented are also beneficial for more mainstream Intel Xeon platforms), we describe some relevant details of KNL here. Figure 3 depicts the high-level organization of the processor. The KNL CPU is organized into up to 36 tiles, connected via a 2D mesh interconnect. The mesh is a departure from the ring architecture employed in Broadwell and previous generation Intel Xeon processors and provides improved scalability because of better on-chip bandwidth and reduced latency. Each tile consists of two compute cores (each with four hardware threads), which share a 1 MB L2 cache. Each core has two VPUs, and because KNL is dual-issue, a VPU can be saturated from a single thread.

A particularly noteworthy feature of KNL is the presence of up to 16 GB of multi-channel DRAM (MCDRAM), a special high-bandwidth memory that sits on the processor package. MCDRAM provides roughly 5× the bandwidth that standard DDR4 SDRAM (double data rate fourth-generation synchronous dynamic random-access memory) can deliver, and it can sustain up to 490 GB/s of memory bandwidth on the well-known STREAM Triad benchmark. The trend over the past decade has seen dramatic increases in peak FLOP rates without commensurate increases in memory bandwidth, but because many operations in scientific computing applications are memory bandwidth-intensive, many of these codes operate in a bandwidth-limited regime and are able to use only a small percentage of the available FLOP/s. Effective utilization of the high-bandwidth memory is critical for achieving good performance on KNL for many applications, including our clustering code.

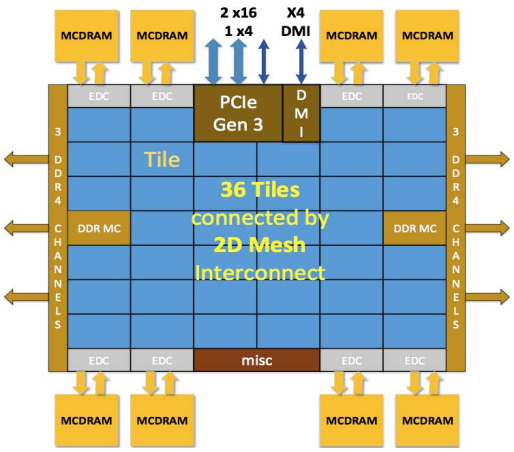


Fig. 3: Block diagram (after [16]) of the second-generation Knights Landing (KNL) Intel Xeon Phi CPU. Note that the maximum number of active tiles is 36, but 38 tiles are shown; the extra tiles are for yield recovery in the manufacturing process.

IV. Benchmarking Setup and Baseline Performance

A. Platform descriptions and configuration

We use three different platforms—to which we will refer by the three-letter abbreviations corresponding to the Intel code names of their CPUs—for our performance benchmarking experiments. *pKluster* is designed for running large clustering problems on distributed memory machines, but since the optimizations we explore in this paper target on-node performance, we present results from experiments using only a single compute node. For problems of sufficient size to provide an appropriate amount of local work per node, our performance optimizations provide similar benefit when running problems spanning multiple nodes. Table I summarizes the characteristics of the three types of compute nodes we use. One system is a 68-core KNL node, and the others are dual-socket systems with "EP" versions of the current (Skylake, SKX) and previous (Broadwell, BDW) generation Intel Xeon server processors. All of the systems have similar power envelopes, so, from a power efficiency standpoint, a comparison of their performance is appropriate. The SKX system has some characteristics similar to KNL features that are new to the Xeon line: it has 512-bit vector registers and uses a variant of the AVX-512 instruction set, and it uses a mesh-on-die interconnect instead of the ring architecture used in previous Xeon generations. Both of the Xeon platforms deliver much higher per-thread performance than does KNL. Therefore, in general it is critically important for applications to possess sufficient parallel scalability to use most or all of the available cores or hardware threads in order to deliver competitive performance on KNL.

The KNL processor has two important configuration options that can be specified at boot time. The first is the *cluster*

TABLE I: Characteristics of the computing platforms used for performance benchmarking in this study.

	Intel Xeon E5-2697 v4	Intel Xeon Gold 6148	Intel Xeon Phi 7250
Code Name	Broadwell (BDW)	Skylake (SKX)	Knights Landing (KNL)
Sockets	2	2	1
Cores	36	40	68
Threads	72	80	272
CPU clock	2.3 GHz	2.4 GHz	1.4 GHz
High-bandwidth memory	-	-	16 GB
DRAM	128 GB @ 2400 MHz	192 GB @ 2666 MHz	98 GB @ 2400 MHz
Instruction set architecture	AVX2	AVX-512F,DQ,CD,BW,VL	AVX-512F,PF,ER,CD
Theoretical peak flops (FP32 / FP64)	2649 / 1324	6144 / 3072	6092 / 3046

mode, which affects the affinity between processor tiles, the cache tag directory, and main memory. We configure our KNL in `quadrant` mode, in which the CPU is divided into four virtual quadrants and addresses are hashed to a directory in the same quadrant as the memory. This mode provides good latency and high bandwidth in a software-transparent manner (that is, without requiring partitioning the memory across four NUMA nodes). The second option is the MCDRAM mode, which affects how the MCDRAM is exposed. From an application programmer's point of view, the simplest MCDRAM mode is `cache` mode, in which the entire MCDRAM is treated as a single, very large direct-mapped cache. We choose to configure our KNL node in `flat` mode, however, in which MCDRAM is exposed as a single NUMA node, because this mode offers the highest performance. In our experiments, we bind all memory requests to this NUMA node via the `numactl` command-line tool.

We use the 2017 Intel C compiler with optimization level `-O3` and build binaries using the highest-level instruction set flags available on each platform, namely, `-xCORE-AVX2`, `-xCORE-AVX512`, `-xMIC-AVX512` on BDW, SKX and KNL, respectively. We experimented with Intel compiler intrinsics to explicitly produce vectorized versions of the most computationally expensive routines, but found that this is not necessary: the Intel compiler's auto-vectorizer generates SIMD instructions with the assistance of a few `#pragma simd` directives we have inserted around the hot loops.

B. Benchmark problem: GSMNP LiDAR classification

In our performance benchmarking experiments, we cluster a data set from a study described in [9]. This data set comes from airborne multiple return Light Detection and Ranging (LiDAR) surveys of the Great Smoky Mountains National Park [17] that straddles the border between Tennessee and North Carolina. LiDAR enables large scale remote sensing of topography, built infrastructure, and vegetation structure. Raw LiDAR data are in the form of point clouds. In this study, k-means clustering of LiDAR point cloud data was used to construct vertical density profiles to characterize vertical vegetation structure. Figure 4 displays the vegetation cover maps and structure class prototypes (centroids) generated using $k = 30$. 30 m $\times$ 30 m horizontal and 1 m vertical (extending to a height of 75 m) spatial resolution was used, with the input data set consisting of 3,186,679 observations, each of 74 variables, requiring 900 MB of storage in single precision.

We note that we have also tested our code with data sets from remotely sensed vegetation phenology [18] and global climate regime classification applications [19], but, due to space considerations, we choose to present results from the GSMNP LiDAR set only. We believe that performance experiments with this data set are representative of the performance trends to be observed when clustering many other geospatial data sets of interest to us, because the data sets have similar dimensionality and consist of continuous variables following Gaussian distributions. Our experiments with other geospatial data sets available to us have been consistent with the performance trends reported here.

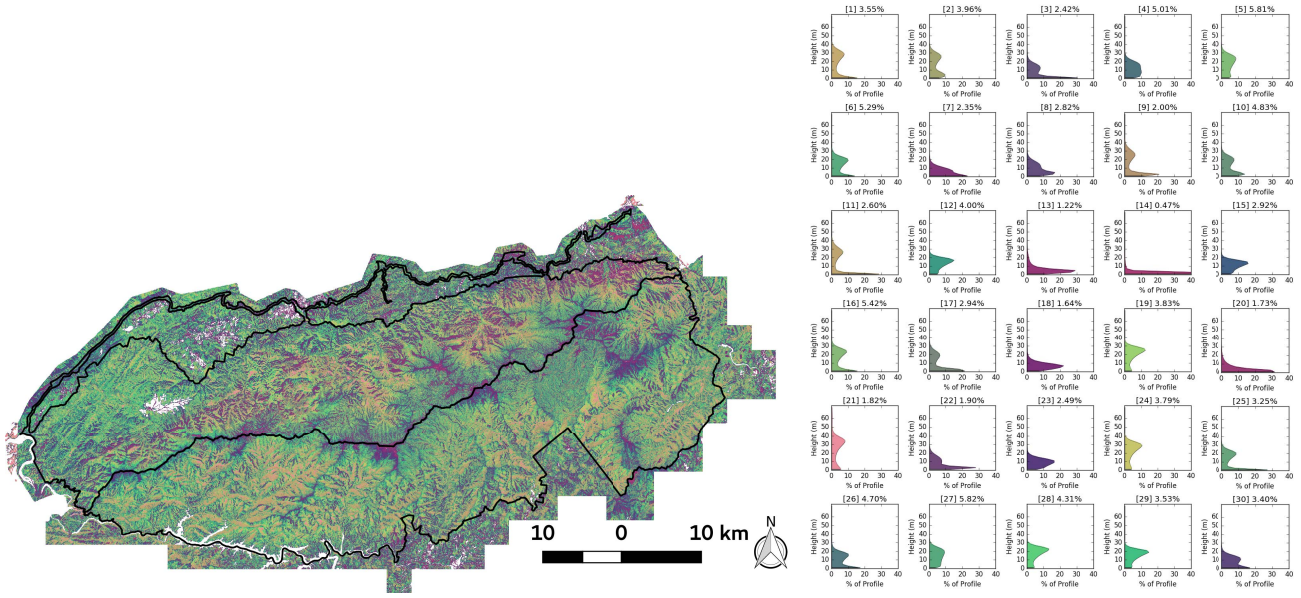
C. Baseline performance

To establish a baseline against which to evaluate the algorithmic improvements and code optimizations made in this study, we ran some initial performance tests using a recent version of *pKluster*, which incorporates the triangle inequality-based "acceleration" algorithm but no further code improvements. Figure 5 summarizes the baseline performance for clustering the GSMNP LiDAR data set with $k = 2000$, which is the highest realistic value of k for this application and best illustrates overall differences in performance for a given platform and implementation. We used one MPI rank per core on each test platform. Compared to BDW, SKX exhibits a 1.3$\times$ speedup. KNL, however, shows surprisingly poor performance, displaying a 2.2$\times$ slowdown versus BDW.

V. Optimizations for Multicore/Manycore CPUs

A. OpenMP threading for effective use of hyperthreads

When a pure MPI approach (with one MPI rank per core) is used, performance of the accelerated k-means clustering approach is surprisingly poor on the KNL processor. Attempts to use more of the available hardware threads by using multiple MPI ranks per core slightly decrease the time in the actual clustering calculation, but increase the total time due to greater overhead in master-worker coordination between the MPI ranks. These results suggest that using more available hardware threads can improve performance on KNL, if we can avoid increasing master-worker overhead. To increase the number of streams of execution without employing additional MPI processes, we have introduced OpenMP threading



(a) Geographic distribution of vegetation classes

(b) Canopy prototypes from $k = 30$ clustering

Fig. 4: Vegetation structure classes and their distribution in the Great Smoky Mountains National Park (GSMNP) derived from k-means clustering ($k = 30$) of the GSMNP LiDAR data set. The spatial distribution is shown at the left, and the prototype canopy structures (cluster centroids) are shown at the right. The color scheme on the map correspond to the colors of the prototypes.

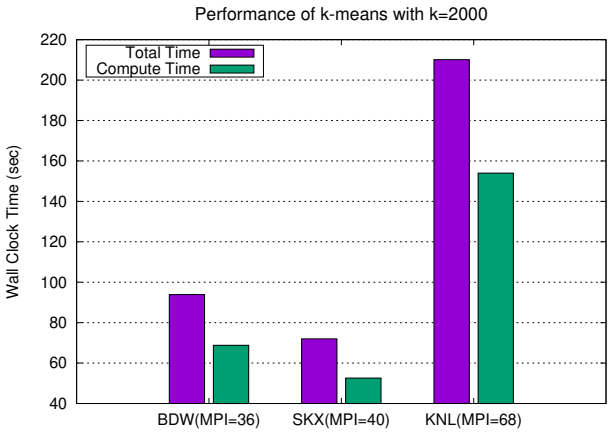


Fig. 5: Baseline performance of the three benchmarking platforms for computing $k = 2000$ clusters for the GSMNP LiDAR data set using the accelerated k-means algorithm.

in the most time-intensive routine, `cluster_aliquot()`, which computes the distances between observations and centroids and then assigns each observation to the cluster with the nearest centroid. We have also introduced OpenMP threading in the intercentroid distance calculation routine, `fill_distance_matrix()`, which is called when using the triangle inequality-based acceleration scheme.

Because using the accelerated k-means algorithm results in some inherent load imbalance—since many observations in an aliquot assigned to a thread might require more point-to-centroid distance calculations compared with its peers—we have explored the various loop scheduling strategies available in OpenMP. The `static` strategy (the default in many OpenMP implementations) assigns all loop iterations to threads in equal chunks upon entry to a parallel loop. This results in the lowest scheduling overhead but may be undesirable when loop iterations differ in their computation cost. For such cases, some variant of dynamic scheduling—in which an interal work queue is used to dynamically distribute loop iterations to threads—may offer higher performance, despite the possibility of significantly higher scheduling overhead. Figure 6 shows that this is the case for our accelerated k-means implementation, with all non-`static` schedules providing approximately 1.4× performance improvement for the $k = 2000$ case, with `dynamic` providing marginally better performance than the `guided` and `auto` options. With our addition of OpenMP threading, we can now use all 272 hardware threads available on the KNL CPU, with the best performance observed when pinning one MPI process to each KNL tile and spawning 8 threads per process (4 threads per core). Using all the available hardware threads through the hybrid MPI-OpenMP approach enables effective utilization of VPU/FMA units and MCDRAM memory, and at the same time reduces the communication bottlenecks on MPI rank 0, resulting in performance improvement of up to 2.8× in the clustering calculation and 2.7× in overall wall-

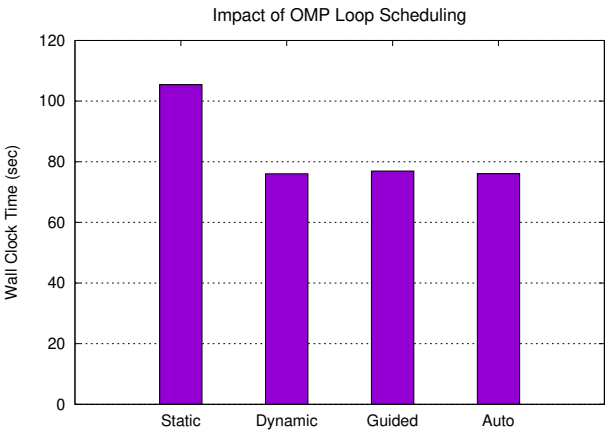


Fig. 6: Impact of different kinds of OpenMP loop scheduling when finding $k = 2000$ clusters of the GSMNP data set on the KNL platform. Setting `schedule` to `dynamic`, `guided`, or `auto` provides significant speedup over the `static` default.

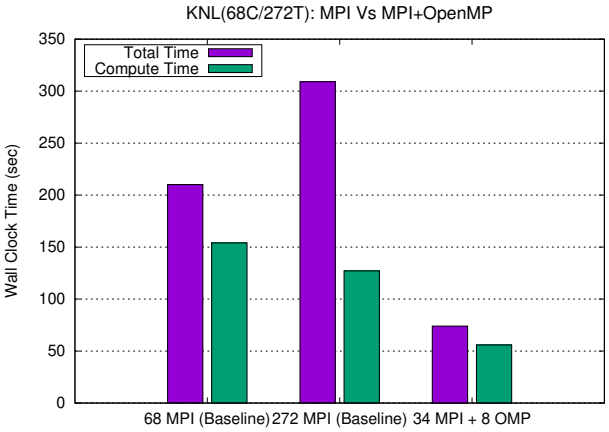


Fig. 7: Comparison of times to cluster the GSMNP LiDAR data set with $k = 2000$ on the KNL processor for different numbers of MPI ranks and OpenMP threads. Using all 272 hyperthreads significantly benefits performance, with 34 MPI processes each using 8 OpenMP threads configuration showing best performance.

clock time (Figure 7). Although using a combination of MPI processes and OpenMP threads is most important on the KNL system because its relatively lightweight cores result in higher communication latency between the MPI processes, we find that the combination is also beneficial on the two Intel Xeon systems that support two logical threads per core, though the performance improvement is less dramatic. As shown in Figure 8, the hybrid MPI-OpenMP implementation delivers 1.3× and 1.4× improvements over pure MPI implementation on BDW and SKX, respectively.

B. Improving computational intensity using level 3 BLAS

Although the addition of OpenMP threading greatly improves the performance of the KNL platform, this performance still only roughly matches that observed on the BDW platform. As the theoretical peak performance of KNL is much higher than that of BDW, and the calculation consists of operations on numerical arrays that the compiler easily vectorizes, it seems that it should be possible to further improve the KNL performance. Achieving good performance on the KNL CPU is strongly dependent on achieving good utilization of the two VPUs present on each KNL core. Although vectorized code is generated for the clustering calculations, the arithmetic intensity—the ratio of floating-point operations to memory loads and stores—is not high enough to keep the VPUs busy, as they must spend too much time waiting for requests to be fulfilled by the memory subsystem. Fortunately, we can achieve greater computational intensity of the observation–centroid distance calculations by expressing the calculation in matrix form: For observation vector x_i and centroid vector z_j, the squared distance between them is $D_{ij} = \|x_i - z_j\|^2$. Via binomial expansion,

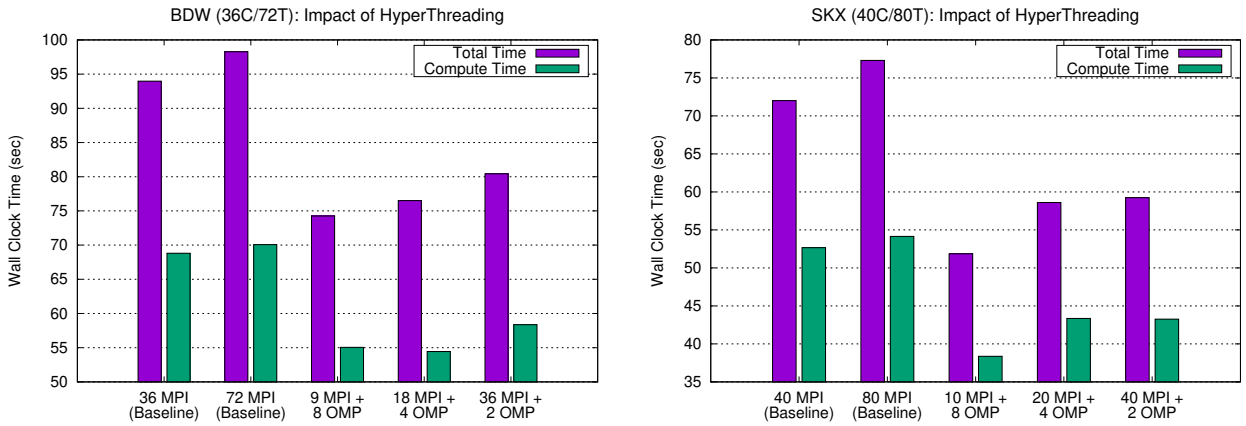
$$D_{ij} = \|x_i\|^2 + \|z_j\|^2 - 2x_i \cdot z_j. \tag{1}$$

The matrix of squared distances can thus be expressed as

$$D = \overline{x}\mathbf{1}^\intercal + \mathbf{1}\overline{z}^\intercal - 2X^\intercal Z, \tag{2}$$

where X and Z are matrices of observations and centroids, respectively, stored in columns, $\overline{x}$ and $\overline{z}$ are vectors of the sum of squares of the columns of X and Z, and $\mathbf{1}$ is a vector of all 1s. The above expression can be calculated in terms of a level-3 BLAS operation (xGEMM), followed by two rank-one updates (xGER, a level-2 operation). Because xGEMM operations admit efficient cache-blocking schemes, a good xGEMM implementation can achieve a very high arithmetic intensity, and hence good utilization of the VPUs. This is desirable for all three of our test platforms, but is particularly critical on KNL, which relies heavily on its many VPUs with wide SIMD lanes for performance. We use the highly optimized BLAS implementations from Intel's Math Kernel Library (MKL) in the experiments presented here. For all but the smallest of matrix dimensions, distance calculations using the above matrix formulation will generally be faster than the straightforward loop over vector distance calculations, and dramatic speedup can be achieved when many observation–centroid distances must be computed.

Figure 9 compares, for varying k, the performance of the accelerated k-means algorithm (using the MPI + OpenMP implementation) and the algorithm employing the matrix formulation on the KNL and BDW platforms. Although the matrix formulation performs many more distance calculations, the efficiency of xGEMM operations on KNL is so high that it outperforms the acceleration scheme for all values of k; it also shows the slowest growth in cost as k increases. On BDW, the story is different: Though the matrix formulation is more efficient in distance comparisons, using it can only speed



(a) Intel Xeon E5-2697 v4 ("Broadwell")

(b) Intel Xeon Gold 6148 ("Skylake")

Fig. 8: Comparison of times to cluster the GSMNP LiDAR data set with $k = 2000$ on the Broadwell (BDW) and Skylake (SKX) Xeon processors for different numbers of MPI ranks and OpenMP threads.

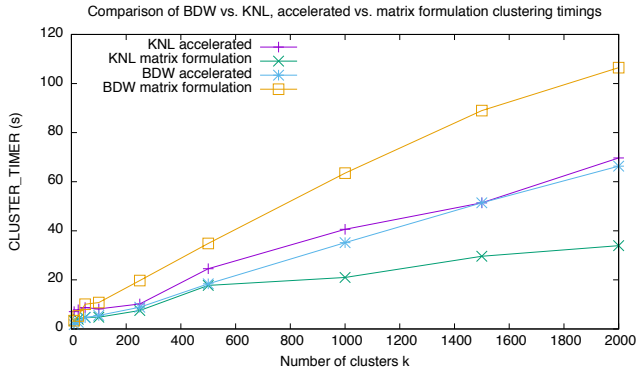


Fig. 9: Comparison of timings for clustering the GSMNP LiDAR dataset for different values of k on the KNL and BDW platforms using the accelerated k-means algorithm and the matrix formulation that uses level-2 and level-3 (xGEMM) BLAS calls.

up the initial iterations (when many distance comparisons are required); after that, the acceleration technique results in dramatically faster iterations. We have not plotted performance of the SKX platform because we did not collect data for all values of k, but at $k = 2000$ (see Figure 10), the accelerated algorithm also delivers the best performance, though the relative difference in performance between the matrix and accelerated version of the algorithm is smaller, which is consistent with the improved xGEMM performance on SKX compared to BDW.

VI. Summary and Future Directions

We have adapted the *pKluster* code used for k-means clustering of geospatial data to enable it to better utilize many compute cores and vector processing units, and demonstrated performance improvements on a true "manycore" CPU (the Intel Knights Landing Xeon Phi), as well as two modern, high core-count server processors employing AVX2 and AVX-512 vector instruction sets (Broadwell and Skylake generation Intel Xeon processors, respectively). Figure 10 summarizes performance improvements we have made: 1.3×, 1.4× and 3.6× over baseline *pKluster* implementation on BDW, SKX and KNL respectively. The addition of OpenMP threading has allowed the code to significantly reduce the MPI master-worker bottleneck when fully subscribing all available hardware threads, which dramatically improves the performance on KNL and also yields appreciable speedup on the BDW and SKX platforms. Use of OpenMP dynamic scheduling also helps to smooth load imbalance that arises naturally in the accelerated k-means algorithm. The addition of a matrix-based algorithmic formulation for the observation–centroid distance comparisons to allow the use of highly optimized level 3 BLAS operations greatly improves arithmetic intensity and enables efficient utilization of vector processing units, which is especially important for KNL.

Although the addition of OpenMP threading significantly reduces the cost of using a master-worker model to coordinate work between the MPI processes, the overhead is still significant on a platform like KNL. In the future, we may reimplement a fully distributed, masterless approach in *pKluster*. We may also investigate a hybrid approach that combines the accelerated k-means method with the matrix formulation for calculation of the observation-centroid distances. In later iterations, the centroids that must be compared against may become fairly predictable, so it may be feasible to do most of the necessary comparisons using a matrix-based approach, with only a few, additional pointwise comparisons required per iteration.

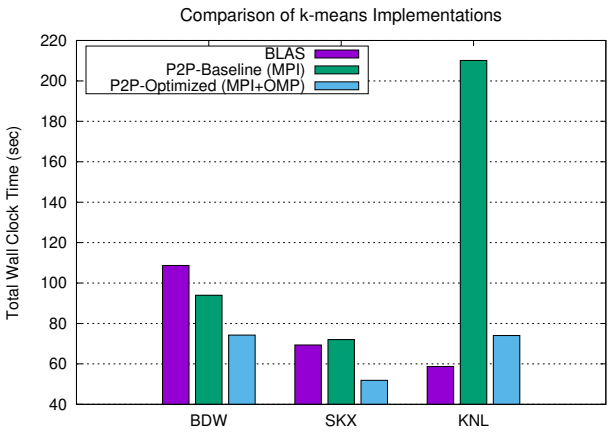


Fig. 10: Performance comparison of k-means implementations for the GSMNP LiDAR dataset with $k = 2000$. Here P2P refers to the *pKluster* implementation using triangle inequality-based acceleration, and BLAS refers to the matrix formulation of distance computations.

VII. Acknowledgments

R. T. Mills was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. JK and FMH were partially supported by the Next Generation Ecosystem Experiments - Arctic (NGEE Arctic) project, which is sponsored by the Terrestrial Ecosystem Sciences (TES) Program, and the Reducing Uncertainties in Biogeochemical Interactions through Synthesis and Computation Scientific Focus Area (RUBISCO SFA), which is sponsored by the Regional and Global Model Analysis (RGMA) Program. The TES and RGMA Programs are activities of the Climate and Environmental Sciences Division (CESD) of the Office of Biological and Environmental Research (BER) in the U.S. Department of Energy Office of Science. WWH, JK, and FMH claim additional support from the Eastern Forest Environmental Threat Assessment Center (EFETAC) in the U.S. Department of Agriculture Forest Service. This manuscript has been authored by UChicago Argonne, LLC under Contract No. DE-AC02-06CH11357 with the U.S. Department of Energy. This manuscript has been co-authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy.

References

[1] J. A. Hartigan, *Clustering Algorithms*. New York: John Wiley & Sons, 1975.

[2] W. W. Hargrove and F. M. Hoffman, "Using multivariate clustering to characterize ecoregion borders," *Comput. Sci. Eng.*, vol. 1, no. 4, pp. 18–25, Jul. 1999.

[3] ——, "Potential of multivariate quantitative methods for delineation and visualization of ecoregions," *Environ. Manage.*, vol. 34, no. Supplement 1, pp. S39–S60, Apr. 2004.

[4] F. M. Hoffman, W. W. Hargrove, D. J. Erickson, and R. J. Oglesby, "Using clustered climate regimes to analyze and compare predictions from fully coupled general circulation models," *Earth Interact.*, vol. 9, no. 10, pp. 1–27, Aug. 2005.

[5] W. W. Hargrove, F. M. Hoffman, and B. E. Law, "New analysis reveals representativeness of the AmeriFlux Network," *Eos Trans. AGU*, vol. 84, no. 48, pp. 529, 535, Dec. 2003.

[6] D. Schimel, W. Hargrove, F. Hoffman, and J. McMahon, "NEON: A hierarchically designed national ecological network," *Front. Ecol. Environ.*, vol. 5, no. 2, p. 59, Mar. 2007.

[7] M. Keller, D. Schimel, W. Hargrove, and F. Hoffman, "A continental strategy for the National Ecological Observatory Network," *Front. Ecol. Environ.*, vol. 6, no. 5, pp. 282–284, Jun. 2008, special Issue on Continental-Scale Ecology.

[8] F. M. Hoffman, J. Kumar, R. T. Mills, and W. W. Hargrove, "Representativeness-based sampling network design for the State of Alaska," *Landscape Ecol.*, vol. 28, no. 8, pp. 1567–1586, Oct. 2013.

[9] J. Kumar, J. Weiner, W. W. Hargrove, S. P. Norman, F. M. Hoffman, and D. Newcomb, "Characterization and classification of vegetation canopy structure and distribution within the Great Smoky Mountains National Park using LiDAR," in *Proceedings of the 15th IEEE International Conference on Data Mining Workshops (ICDMW 2015)*, P. Cui, J. Dy, C. Aggarwal, Z.-H. Zhou, A. Tuzhilin, H. Xiong, and X. Wu, Eds., Institute of Electrical and Electronics Engineers (IEEE). Conference Publishing Services (CPS), Nov. 2015, pp. 1478–1485.

[10] W. W. Hargrove, F. M. Hoffman, and T. Sterling, "The do-it-yourself supercomputer," *Sci. Am.*, vol. 265, no. 2, pp. 72–79, Aug. 2001. [Online]. Available: http://www.sciam.com/article.cfm?articleID=000E238B-33EC-1C6F-84A9809EC588EF21

[11] F. M. Hoffman, W. W. Hargrove, R. T. Mills, S. Mahajan, D. J. Erickson, and R. J. Oglesby, "Multivariate Spatio-Temporal Clustering (MSTC) as a data mining tool for environmental applications," in *Proceedings of the iEMSs Fourth Biennial Meeting: International Congress on Environmental Modelling and Software Society (iEMSs 2008)*, M. Sànchez-Marrè, J. Béjar, J. Comas, A. E. Rizzoli, and G. Guariso, Eds., Jul. 2008, pp. 1774–1781.

[12] J. Kumar, R. T. Mills, F. M. Hoffman, and W. W. Hargrove, "Parallel k-means clustering for quantitative ecoregion delineation using large data sets," in *Proceedings of the International Conference on Computational Science (ICCS 2011)*, ser. Procedia Comput. Sci., M. Sato, S. Matsuoka, P. M. Sloot, G. D. van Albada, and J. Dongarra, Eds., vol. 4. Amsterdam: Elsevier, Jun. 2011, pp. 1602–1611.

[13] S. J. Phillips, "Reducing the computation time of isodata and k-means unsupervised classification algorithms," in *Geoscience and Remote Sensing Symposium, 2002 (IGARSS'02)*, vol. 3, Jun. 2002, pp. 1627–1629.

[14] ——, "Acceleration of k-means and related clustering algorithms," in *ALENEX '02: Revised Papers from the 4th International Workshop on Algorithm Engineering and Experiments*, D. M. Mount and C. Stein, Eds. London, UK: Springer-Verlag, 2002, pp. 166–177.

[15] C. Martin, "Multicore processors: challenges, opportunities, emerging trends," in *Proc. Embedded World Conference*, vol. 2014, 2014, p. 1.

[16] A. Sodani, R. Gramunt, J. Corbal, H. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. Liu, "Knights landing: Second-generation intel xeon phi product," *IEEE Micro*, vol. 36, no. 2, pp. 34–46, Mar 2016.

[17] T. Jordan, M. Madden, B. Yang, J. Sharma, and S. Panda, "Acquisition of LiDAR for the Tennessee Portion of Great Smoky Mountains National Park and the Foothills Parkway," Center for Remote Sensing and Mapping Science (CRMS), Department of Geography, The University of Georgia, Athens, Georgia, USA, Tech. Rep. USGS Contract # G10AC0015, 2011.

[18] R. T. Mills, J. Kumar, F. M. Hoffman, W. W. Hargrove, J. P. Spruce, and S. P. Norman, "Identification and visualization of dominant patterns and anomalies in remotely sensed vegetation phenology using a parallel tool for principal components analysis," *Procedia Computer Science*, vol. 18, pp. 2396 – 2405, 2013, 2013 International Conference on Computational Science. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050913005541

[19] F. M. Hoffman, W. W. Hargrove, J. Kumar, Z. L. Langford, and D. M. Maddalena, "High performance computational landscape ecology and using clustering to define climate regimes," 9th International Association for Landscape Ecology (IALE) World Congress (July 5–10, 2015), Portland, Oregon, USA, July 5-10 2015.