Scalable Tools for Analysis of Massive Remote-Sensing Datasets on High-Performance Computers

Richard Tran Mills, Argonne National Laboratory
Vamsi Sripathi, Intel Corporation
Jitendra Kumar, Oak Ridge National Laboratory
Sarat Sreepathi, Oak Ridge National Laboratory
Forrest M. Hoffman, Oak Ridge National Laboratory
William W. Hargrove, USDA Forest Service Southern Research Station

US-IALE Annual Meeting, Fort Collins, CO
Fort Collins, CO — April 8, 2019

Introduction

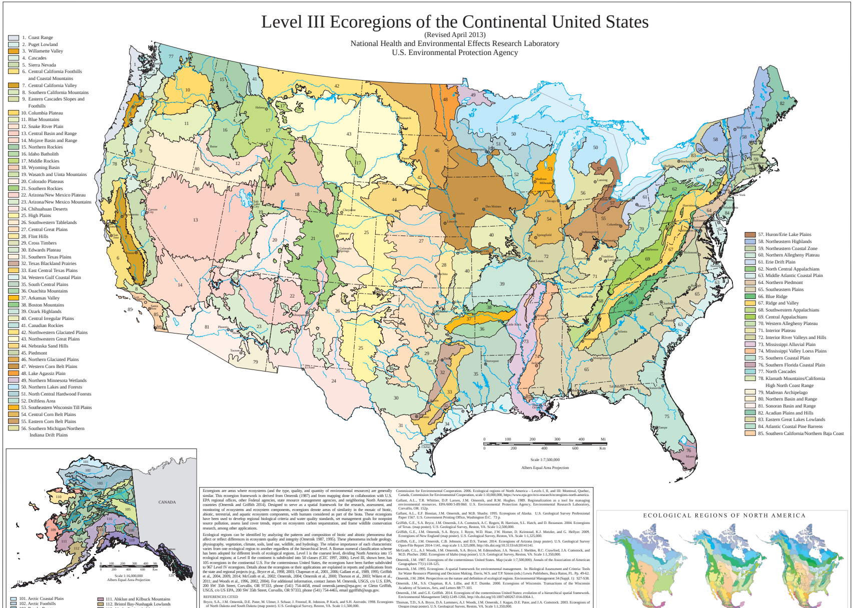
- Increasing availability of high-resolution geospatiotemporal data sets from varied sources:
  - Observatory networks
  - Remote sensing platforms
  - Computational Earth system models
- New possibilities for knowledge discovery and mining of geoscience data sets fused from disparate sources.
- Traditional tools impractical for analysis/synthesis of data sets this large: Need new approaches to utilize complex memory hierarchies and high levels of available parallelism in state-of-the-art high-performance computing platforms.
- We have adapted pKluster—an open-source tool for accelerated *k*-means clustering we use for many geospatiotemporal applications—to effectively utilize state-of-the art multi- and manycore processors, such as the second-generation Intel Xeon Phi ("Knights Landing") processor, as well as GPGPUs.

Scalable *k*-means Clustering with pKluster

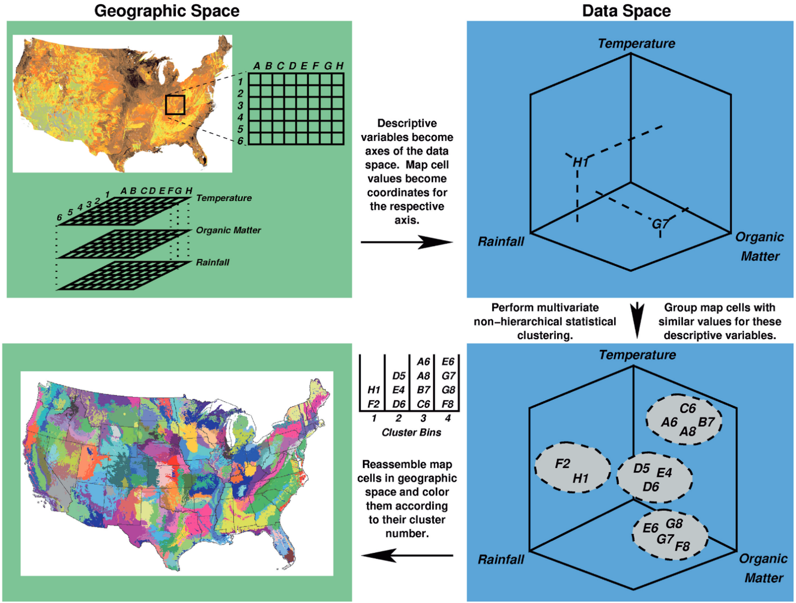
Our distributed-memory clustering code has a long history...

Figure: Originally developed in 1996–1997 for use on the Stone Soupercomputer, a very early Beowulf-style cluster constructed entirely out of surplus parts (see "The Do-It-Yourself Supercomputer", *Scientific American*, 265 (2), pp. 72-79, 2001.)

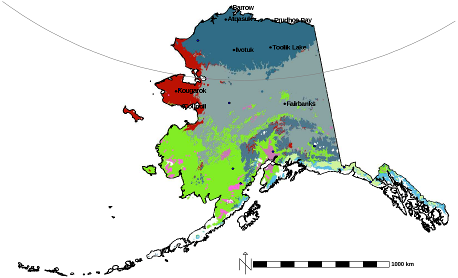
Original motivation: Replacing hand-drawn ecoregionalizations



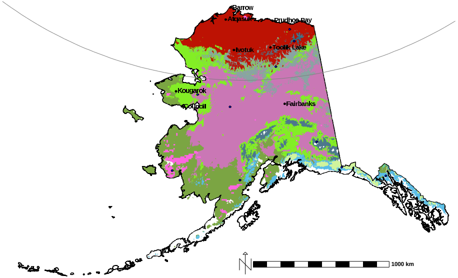
Quantitative Ecoregionalization through Multivariate Spatio(-Temporal) Clustering



Quantitative Ecoregionalization through Time: Sampling Network Design



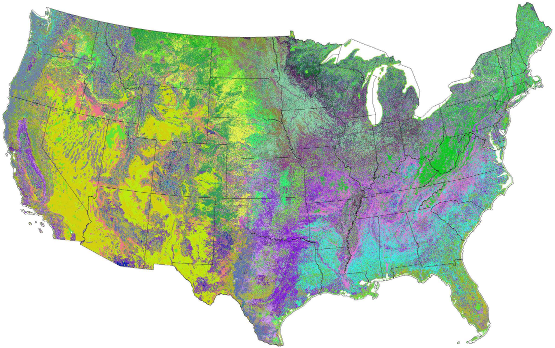
(a) 10 ecoregions, present (2000-2009)

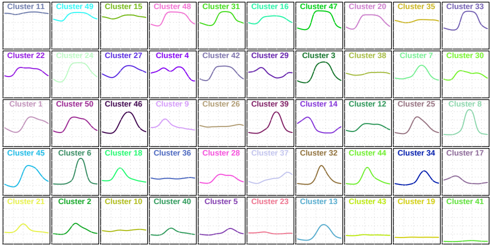


(b) 10 ecoregions, future (2090-2099)

Figure: Geospatiotemporal clustering of a combination of observational data and downscaled general circulation model results projects dramatic shifts in location of Alaska ecoregions using downscaled 4 km GCM results. Arctic tundra projected to be at 0.78% of current extent by 2099. DOI: 10.1007/s10980-013-9902-0.
2014 US-IALE Outstanding Paper in Landscape Ecology.

MODIS NVDI-based phenoregionalization





GSMNP LiDAR-derived canopy structure classification

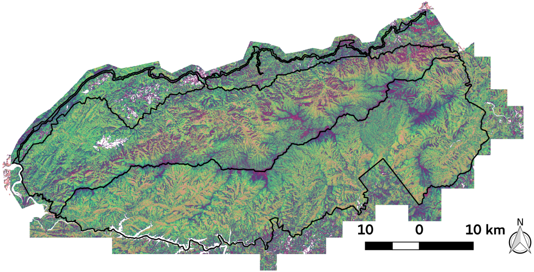
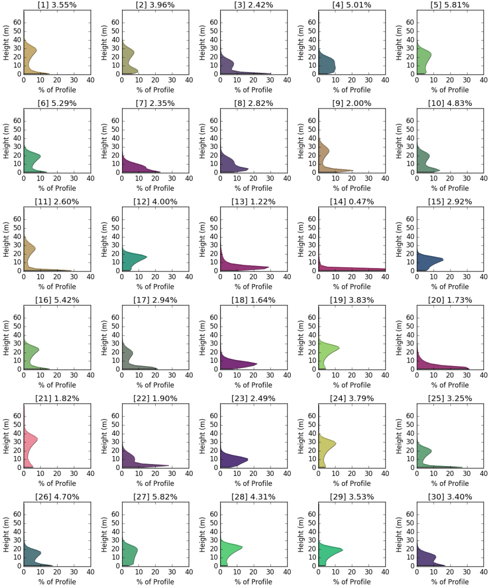


Figure: Map (above) showing the 30 most-different classes of vegetation canopy structure, as identified by k-means clustering (right) for the Great Smoky Mountains National Park.



Scalable *k*-means Clustering with pKluster: Parallel Computers Have Evolved!

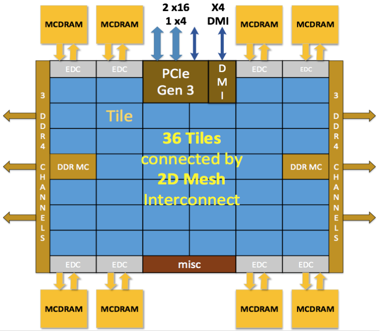
- When pKluster was initially written, on-node parallelism was virtually nonexistent on commodity PCs; focus was purely on distributed-memory parallelism (i.e., Message Passing Interface—MPI).
- Modern HPC compute nodes increasingly feature high degrees of on-node parallelism:
  - Modern CPUs feature large numbers of compute cores, increasing reliance on SIMD (vector instructions).
  - Many new supercomputers are concentrating almost all power in GPU "accelerators".

Figure: O. G.

Figure: New School

Manycore Computing Architectures

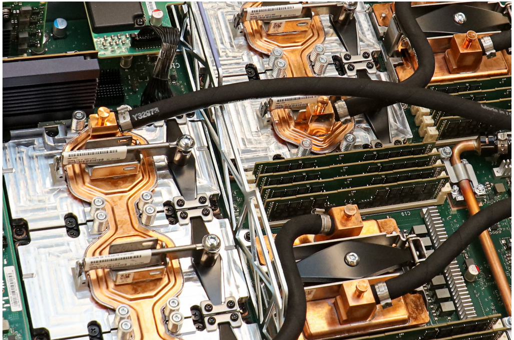
- In recent years, the number of compute cores and hardware threads has been dramatically increasing.
- Seen in GPGPUS, “manycore” processors such as the Intel Xeon Phi, and even on standard server processors (e.g., Intel Xeon Skylake).
- There is also increasing reliance on data parallelism/fine-grained parallelism.
  - Current Intel consumer-grade processors have 256-bit vector registers and support AVX2 instructions.
  - Second-generation Intel Xeon Phi processors and Intel Xeon (Skylake and beyond) server processors have 512-bit vectors/AVX512 instructions.



At left, “Knights Landing” (KNL) Xeon Phi processor:

- Up to 36 tiles interconnected via 2D mesh
- Tile: 2 cores + 2 VPU/core + 1 MB L2 cache
- Core: Silvermont-based, 4 threads per core, out-of-order execution
- Dual issue; can saturate both VPUs from a single thread
- 512 bit (16 `floats` wide) SIMD lanes, AVX512 vector instructions
- High bandwidth memory (MCDRAM) on package: 490+ GB/s bandwidth on STREAM triad[2]
- Powers the NERSC Cori and ALCF Theta supercomputers

OLCF Summit Supercomputer



System totals

- $\sim$ 200 PFlop/s theoretical peak
  143 PFlop/s LINPACK—#1 in TOP500
- 4,608 compute nodes

Node configuration

- Compute:
  - Two IBM Power9 CPUs, each 22 with cores, 0.5 DP TFlop/s
  - Six NVIDIA Volta V100 GPUs, each with 80 SMs–32 FP64 cores/SM, 7.8 DP TFlop/s
- Memory:
  - 512 GB DDR4 memory
  - 96 (6×16) GB high-bandwidth GPU memory
  - 1.6 TB nonvolatile RAM (I/O burst buffer)

Almost all compute power is in GPUs!

CPU Benchmarking Platforms and Problem

Performance benchmarking platforms:

	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

- SKX and KNL double the SIMD width of BDW (256 to 512 bits)
- SKX and KNL have similiar peak flops; KNL more dependent on SIMD and thread parallelism

Benchmark problem: GSMNP LiDAR clustering

- 1.5 million observations
- 74 dimensions
- $k = 2000$ clusters

Parallel k-means clustering algorithm

k-means clustering

Goal: Partition data into k clusters, such that centroid c_j minimizes the total distance $D_j = \sum d(c_j, a)$ to points a in cluster P_j.
Iterative calculation: Given initial partition, find centroid of each cluster and repartition according to closest centroid (essentially Lloyd's algorithm, or voronoi relaxation).

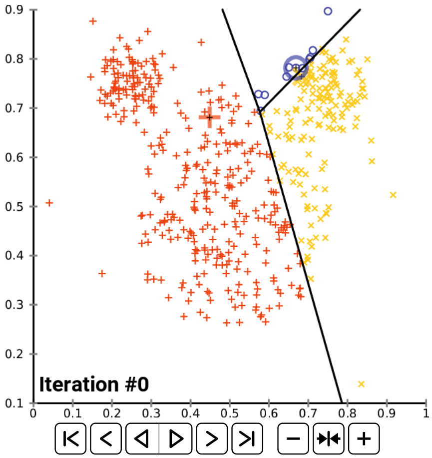


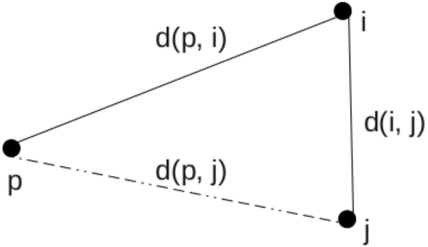
Figure: Illustration of k-means iteration for $k = 3$. https://commons.wikimedia.org/wiki/File:K-means_convergence.gif

Parallel implementation in *pKluster*

- Centralized master-worker paradigm
- Start from some initial centroids (chosen offline)
- Master:
  - Broadcasts centroids and aliquot assignment to workers
  - Collects new cluster assignments from workers
  - Recomputes centroids
- Workers, for an assigned aliquot:
  - Compute observation-to-centroid distances
  - Assign each observation to closest centroid

Accelerated k-means clustering

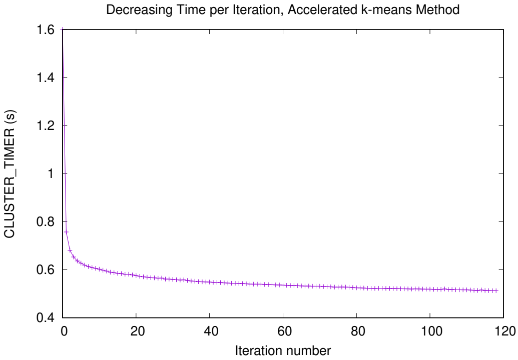
- Classical k-means actually performs far more distance calculations than required!
- Use the triangle inequality to eliminate unnecessary point-to-centroid distance computations based on the previous cluster assignments and the new inter-centroid distances.
- Reduce evaluation overhead by sorting inter-centroid distances so that new candidate centroids c_j are evaluated in order of their distance from the former centroid c_i. Once the critical distance $2d(p, c_i)$ is surpassed, no additional evaluations are needed, as the nearest centroid is known from a previous evaluation.



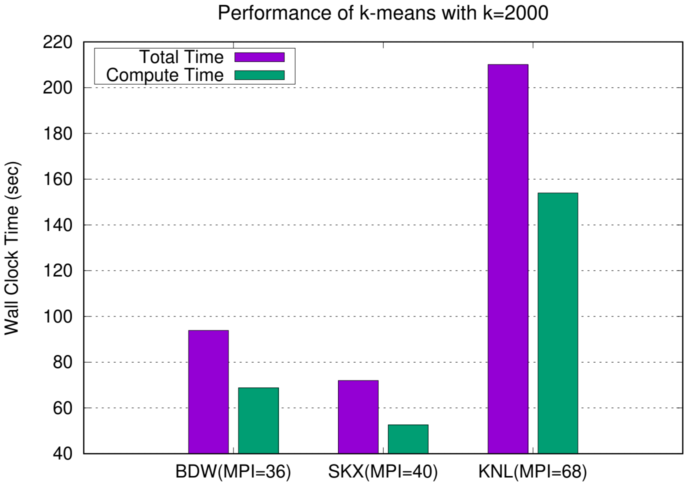
$d(i,j) \leq d(p,i) + d(p,j)$
$d(i,j) - d(p,i) \leq d(p,j)$
if $d(i,j) \geq 2d(p,i)$:
$\quad d(p,j) \geq d(p,i)$
$\quad$ without calculating the distance $d(p,j)$

Accelerated k-means clustering

- Classical k-means actually performs more distance calculations than required!
- Use the triangle inequality to eliminate unnecessary point-to-centroid distance computations based on the previous cluster assignments and the new inter-centroid distances.
- Reduce evaluation overhead by sorting inter-centroid distances so that new candidate centroids c_j are evaluated in order of their distance from the former centroid c_i. Once the critical distance $2d(p, c_i)$ is surpassed, no additional evaluations are needed, as the nearest centroid is known from a previous evaluation.



Baseline (accelerated *k*-means) Performance

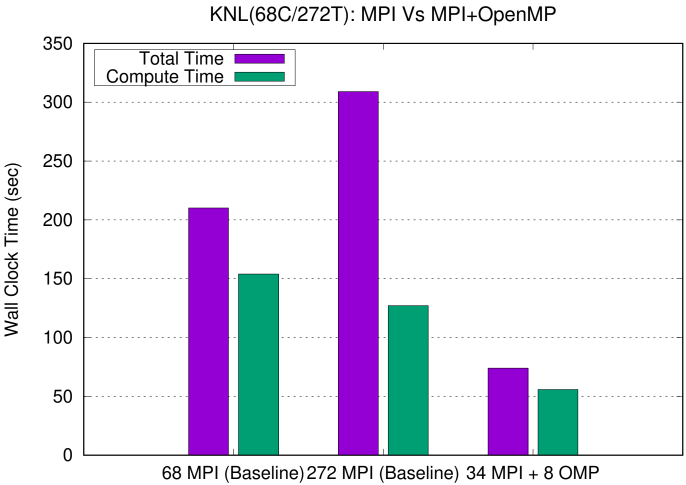


- 1.3X speedup on SKX vs. BDW
- Significant slowdown (2.2X) on KNL vs. BDW

Effective Use of Hyperthreads

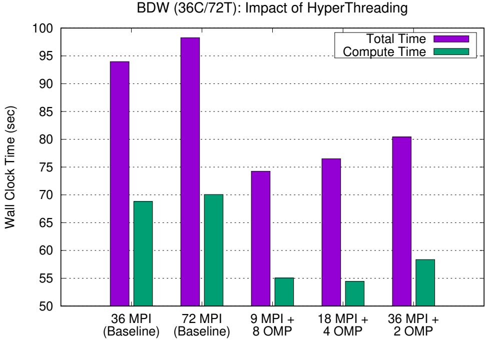
- Using a pure MPI approach (one MPI rank per core), performance of the accelerated k-means clustering approach is surprisingly poor on the "Knights Landing" (KNL) processor.
- Using two MPI ranks per core slightly decreases time in the actual clustering calculation, but slightly increases total time due to greater overhead in master-worker coordination.
- This suggests that using more available hardware threads can improve performance on KNL, if we can avoid increasing master-worker overhead.

Performance Optimizations: OpenMP Parallelism on KNL

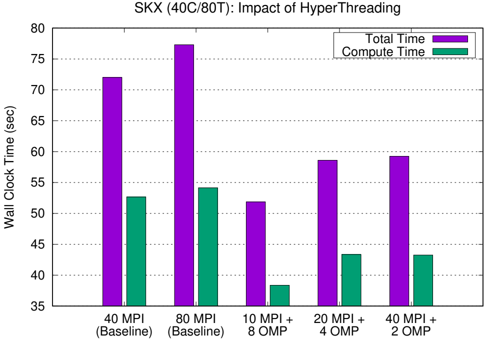


- Hybrid MPI-OpenMP version of distance calculation function effectively utilizes FMA units and reduces the bottleneck on rank 0.
- Use dynamic loop scheduling to smooth load imbalance due to triangle inequality (many observations in an aliquot might skip point-to-centroid distance calculation).
- Pin each MPI to a KNL "tile" and spawn 8 threads (4 threads per core).
- 2.8X improvement.

Performance Optimizations: OpenMP Parallelism on BDW and SKX



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



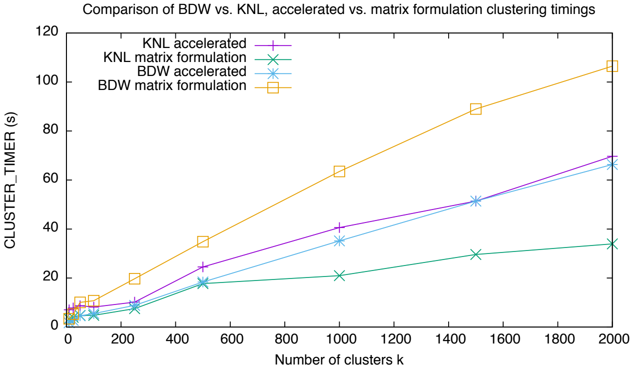
(b) Intel Xeon Gold 6148 ("Skylake")

Figure: 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.

Improving computational intensity

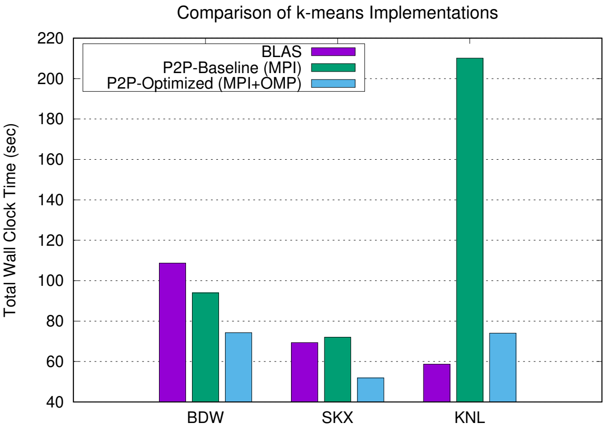
- Can achieve greater computational intensity of the observation–centroid distance calculations by expressing the calculation in matrix form:
  - For observation vector x_i and centroid vector z_j, the squared distance between them is $D_{ij} = \left\|x_i - z_j\right\|^2$.
  - Via binomial expansion, $D_{ij} = \|x_i\|^2 + \left\|z_j\right\|^2 - 2x_i \cdot z_j$.
  - The matrix of squared distances can thus be expressed as $D = \overline{x}\mathbf{1}^\intercal + \mathbf{1}\overline{z}^\intercal - 2X^\intercal Z$, 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.
- 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).
- We use highly optimized BLAS implementations from Intel's MKL and NVIDIA cuBLAS to speed up distance calculations on Xeon Phi and GPGPUs, respectively.
- Distance calculations using above formulation can be dramatically faster than the straightforward loop over vector distance calculations when many distance comparisons must be made.
- Using the matrix formulation for distance comparisons in early k-means iterations is straightforward; a more complicated approach we hope to explore is using the matrix formulation in combination with the acceleration techniques described above, in which only a subset of observation–centroid distances are calculated.

BDW vs. KNL, Accelerated (MPI + OpenMP version) vs. Matrix Formulation



- Though BLAS/matrix formulation performs many more distance calculations, xGEMM is so efficient on KNL that it outperforms acceleration scheme for all k; also shows slowest growth in cost as k increases.
- On BDW, matrix formulation only benefits initial iterations (when many distance comparisons are required); after that, acceleration technique results in dramatically faster iterations.

Performance Improvements Summary

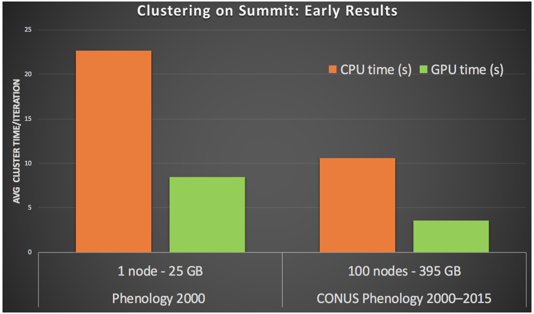


- BLAS formulation yields best performance on KNL, despite many more distance calculations than point-to-point (P2P) approach using "acceleration"; slightly slower then P2P distance calculation on SKX.
- Best performance on SKX with acceleration, though difference between matrix and accelerated algorithm is smaller—consistent with the improved xGEMM performance on SKX compared to BDW
- Overall performance improvements:
  - KNL: 3.5X
  - BDW: 1.3X
  - SKX: 1.4X

Early Summit Benchmarking Results

The matrix formulation for the distance calculations facilitates using GPUs on Summit: Replace BLAS calls with NVIDIA cuBLAS calls.

Problem Config	Data Size	# Clusters	Nodes	CPU time/iter	GPU time/iter	Speedup
Phenology 2000	25 GB	1000	1	22.69 s	8.47 s	2.67
Phenology 2000–2015	395 GB	1000	100	10.60 s	3.59 s	2.95



Future Directions

pKluster software development

- Investigate hybrid approach combining accelerated k-means method and matrix formulation within the same iteration.
- Re-implement a fully distributed, masterless approach in the current version of the code to handle cases in which master-slave overhead is high (e.g., many cases on `KNL`).
- Add support for emerging high-capacity, non-volatile memory technologies.
- Supported open-source release under Apache License 2.0.
- Explore integration with Portable, Extensible Toolkit for Scientific Computation (PETSc), or reimplementation of our algorithms in PETSc.

Complementary machine-learning techniques

- Sophisticated neural networks becoming more accessible
  - High level frameworks like Keras allow easy utilization of libraries such as TensorFlow
  - Consumer-grade GPUs are enabling expensive training even without access to expensive HPC hardware
- Open-source PETSc-based support vector machine (SVM) implementations allow scalable training of SVMs