Scalable Algorithms for Clustering Large Geospatiotemporal Data Sets on Intel Architectures

Collaborators:
Richard T. Mills, Argonne National Laboratory
Sarat Sreepathi, Oak Ridge National Laboratory
Forrest M. Hoffman, Oak Ridge National Laboratory
Jitendra Kumar, Oak Ridge National Laboratory
William W. Hargrove, USDA Forest Service

Vamsi Sripathi
Intel

Outline

- Motivation
- Parallel k-means Clustering
- Intel Computing Architectures
- Baseline Performance
- Performance Optimizations
- Future Trends

Motivation

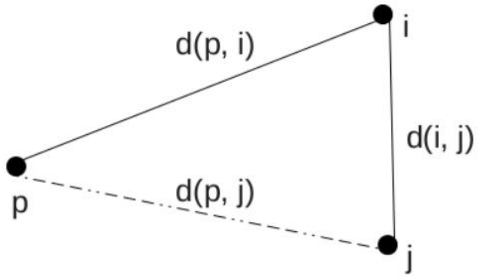
- Rapid proliferation of data in Earth Sciences and other domains
  - Advanced sensors – high fidelity data
  - Remote Sensing Platforms
  - Observational Facilities
- Applications
  - Vegetation Mapping and Characterization
  - Development of Eco-regions
  - Species Distribution
- Critical need for **High Performance** Big Data Analytics

Parallel k-means Clustering

- Centralized Master-Worker paradigm
- Pick initial centroids
- Workers
  - Compute observation-to-centroid distances
  - Update centroids and cluster assignments
- Dataset
  - # of Observations = 1.5 million
  - # of Co-ordinates = 74
  - # of Clusters = 2000

Accelerated k-means: Triangle Inequality

- Implemented an accelerated version of the k-means process using two techniques described by Phillips (doi:10.1109/IGARSS.2002.1026202)
- Use triangle inequality principle 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 $2 * d(p, C_i)$ is surpassed, no additional evaluations are needed, as the nearest centroid is known from a previous evaluation



$d(i,j) \leq d(p,i) + d(p,j)$

$d(i,j) - d(p,i) \leq d(p,j)$

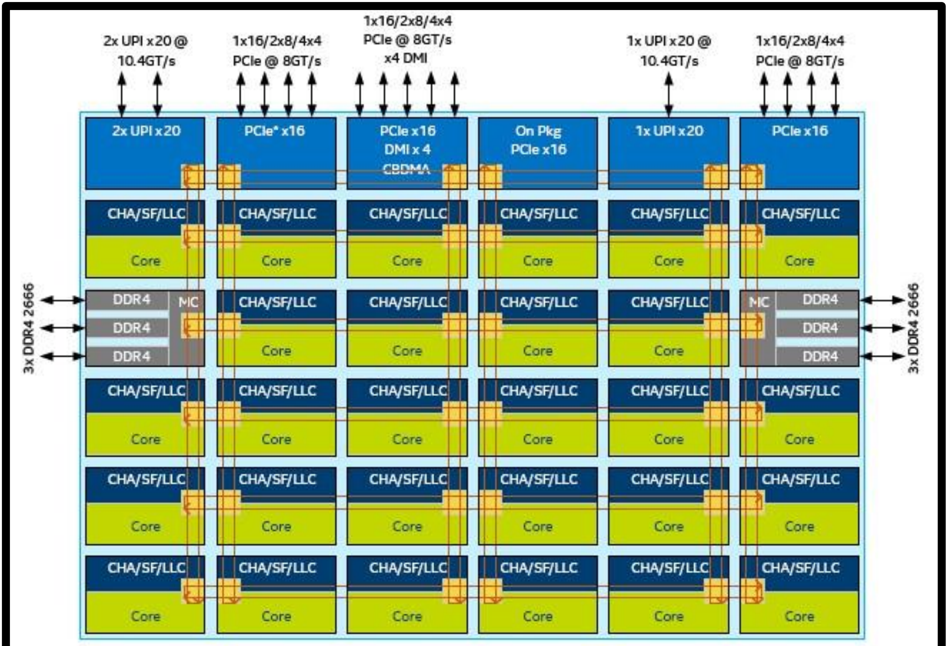
if $d(i,j) \geq 2d(p,i)$:

  $d(p,j) \geq d(p,i)$

  without calculating the distance

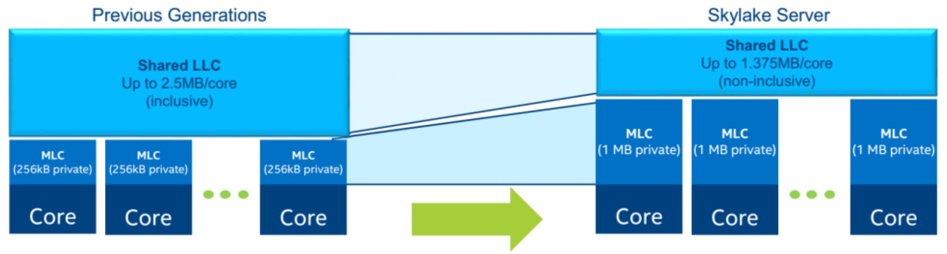
$d(p,j)$

Intel® Xeon® - Skylake

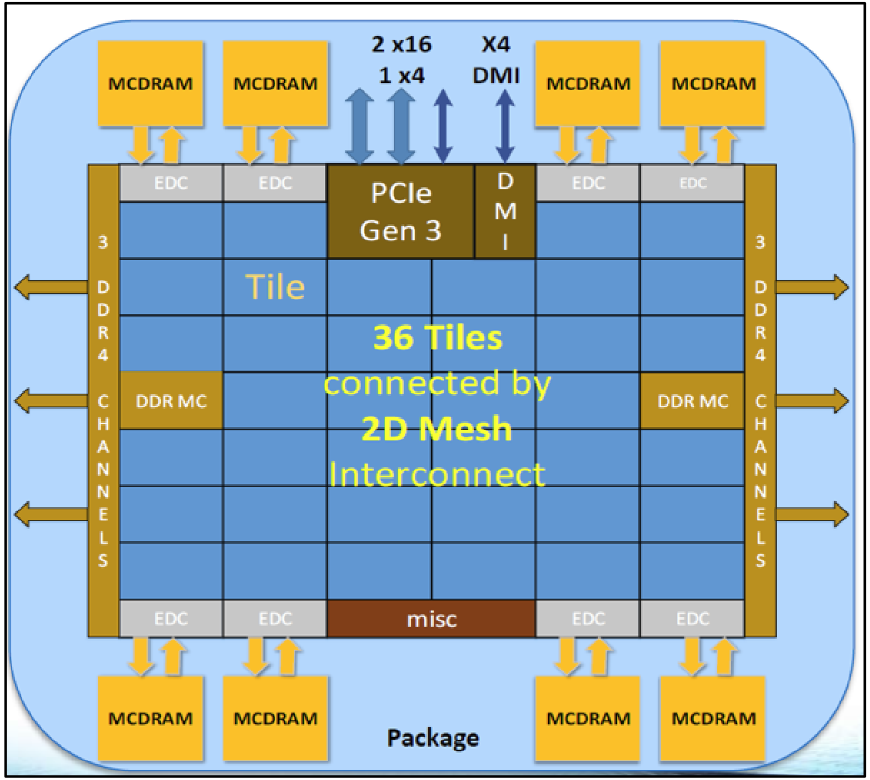


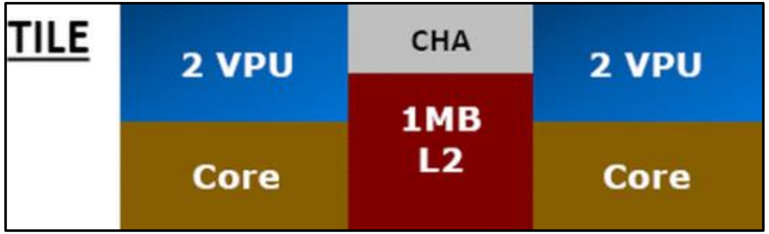
CHA – Caching and Home Agent ; SF – Snoop Filter; LLC – Last Level Cache;
Core – Skylake-SP Core; UPI – Intel® UltraPath Interconnect

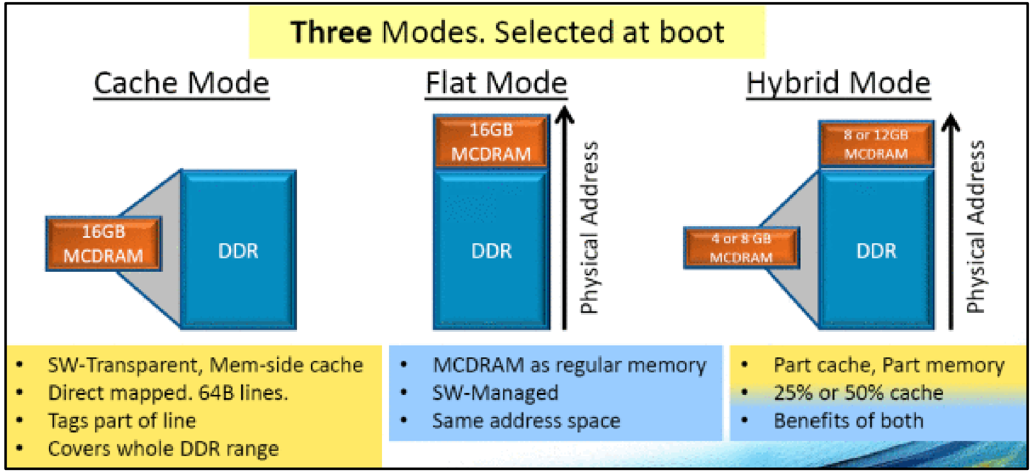
Features	Intel® Xeon® E5-2600 v4	Intel® Xeon® (Skylake-SP)
Cores Per Socket	Up to 22	**Up to 28**
Threads Per Socket	Up to 44 threads	**Up to 56 threads**
Last-level Cache (LLC)	Up to 55 MB	**Up to 38.5 MB (non-inclusive)**
QPI/UPI Speed (GT/s)	2x QPI channels @ 9.6 GT/s	**Up to 3x UPI @ 10.4 GT/s**
PCIe* Lanes/ Controllers/Speed(GT/s)	40 / 10 / PCIe* 3.0 (2.5, 5, 8 GT/s)	**48 / 12 / PCIe 3.0 (2.5, 5, 8 GT/s)**
Memory Population	4 channels of up to 3 RDIMMs, LRDIMMs, or 3DS LRDIMMs	**6 channels of up to 2 RDIMMs, LRDIMMs, or 3DS LRDIMMs**
Max Memory Speed	Up to 2400	**Up to 2666**
TDP (W)	145 - 55	**205 - 70**



Intel® Xeon® Phi - Knight Landing





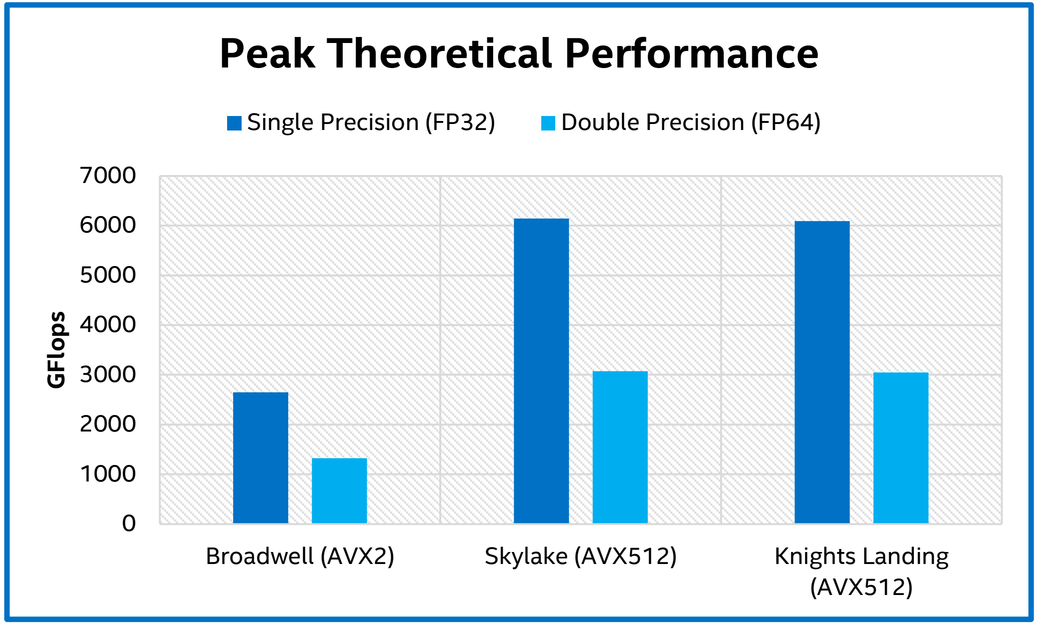


STREAM Traiad (GB/s) : MCDRAM (400+), DDR (90+)

Benchmarking Platforms

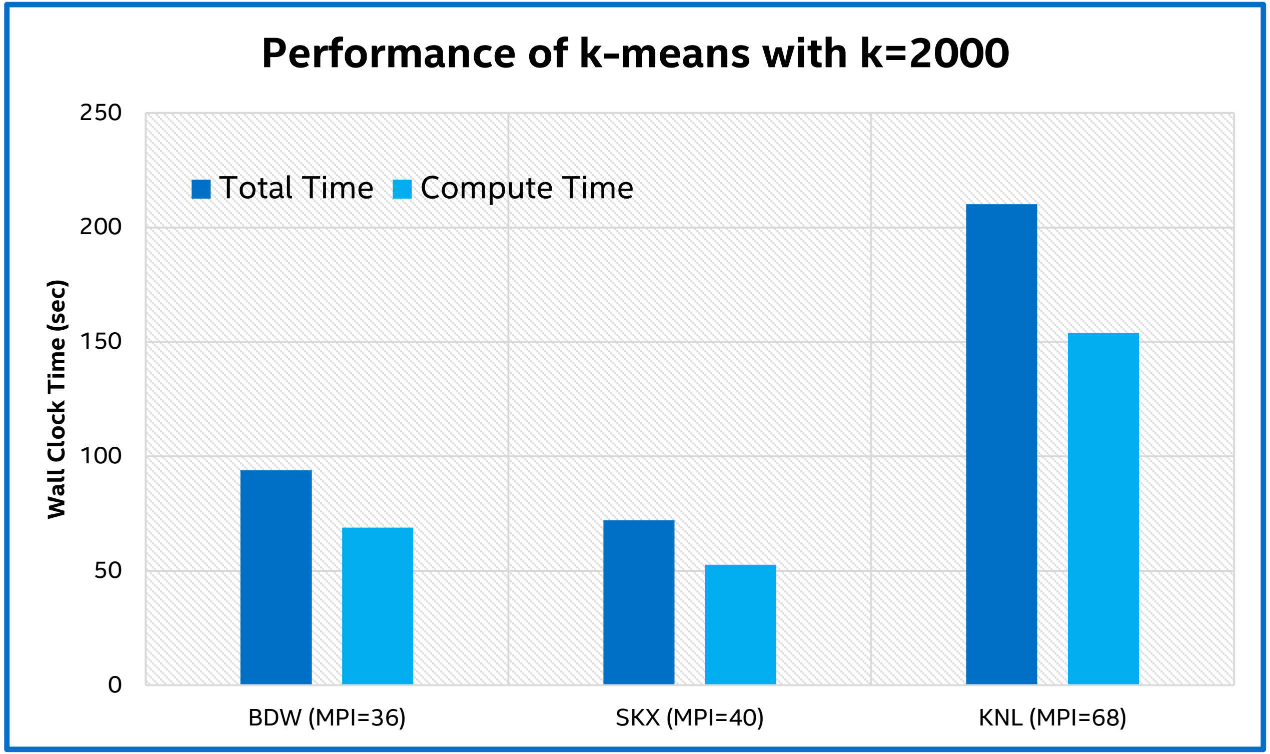
	Intel(R) Xeon(R) CPU E5-2697 v4	Intel(R) Xeon(R) Gold 6148	Intel(R) Xeon Phi(TM) CPU 7250
Code Name	Broadwell (BDW)	Skylake (SKX)	Knights Landing (KNL)
Sockets	2	2	1
Cores	36	40	68
Threads (HT enabled)	72	80	272
CPU Clock (GHz)	2.3	2.4	1.4
HBM	-	-	16 GB
Memory	128 GB @ 2400 MHz	192 GB @ 2666 MHz	98 GB @ 2400 MHz
ISA	AVX2	AVX512{F, DQ, CD, BW, VL}	AVX512{F,PF, ER, CD}

AVX2 Vs AVX512F



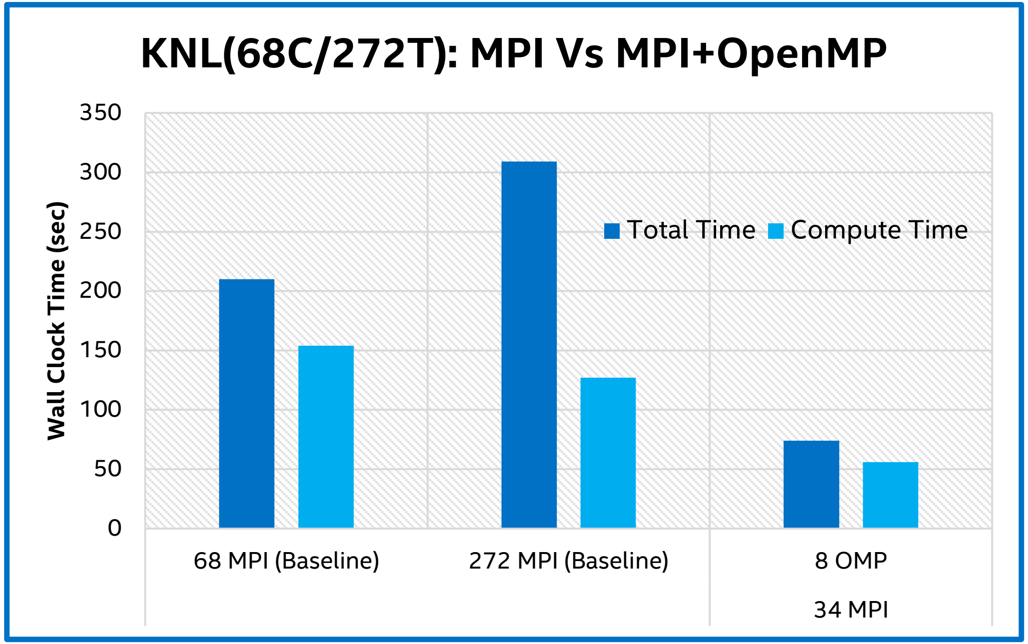
		AVX2	AVX512
Vector Register Length		256 bits	512 bits
# of FMA's per cycle		2	2
Single Precision	# of Elements per register	8	16
	Flops per cycle	32	64
Double Precision	# of Elements per register	4	8
	Flops per cycle	16	32

Baseline Performance



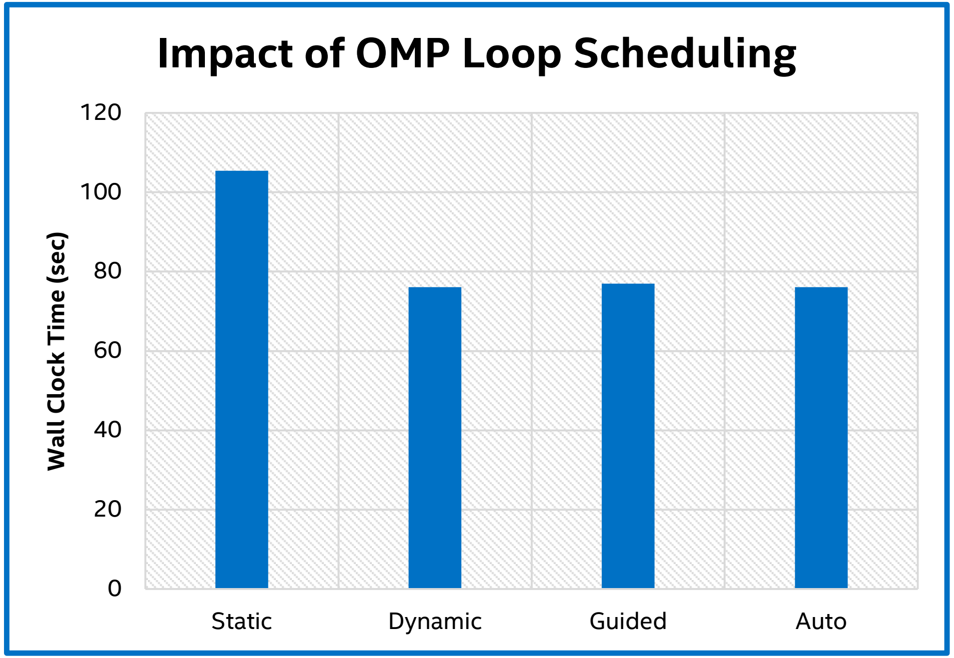
- ❑ 1.3x speed-up on SKX compared to BDW
- ❑ Significant slowdown (2.2x) on KNL

Performance Optimizations: OpenMP Parallelism



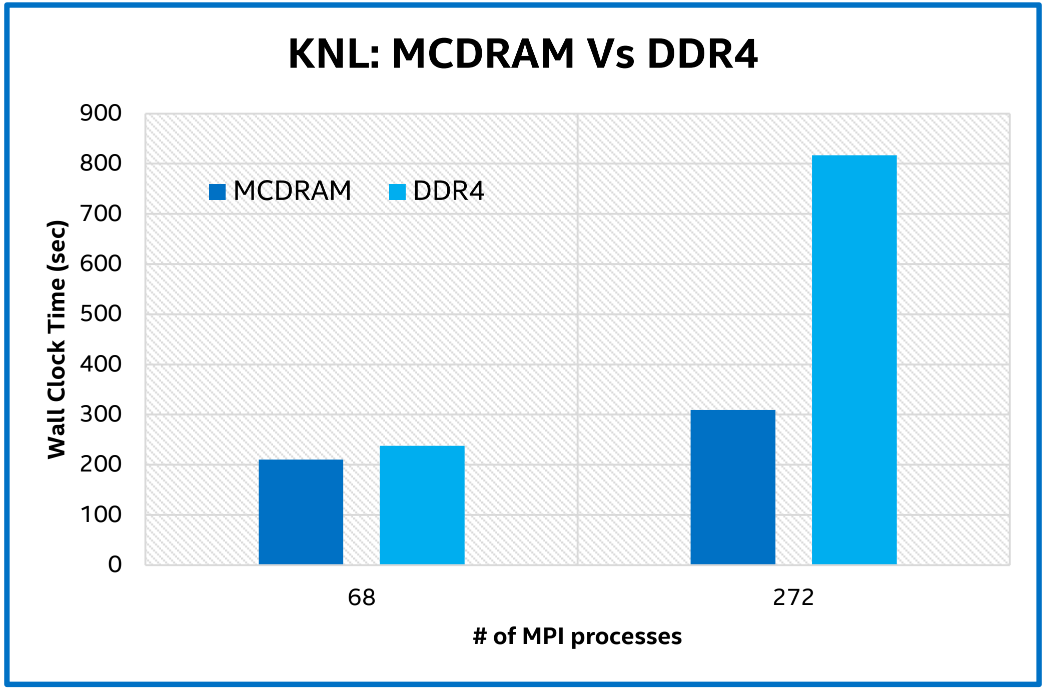
- Developed a hybrid MPI-OpenMP version of distance calculation function to effectively use the FMA units and to reduce the bottleneck on rank-0
- Pin each MPI to a KNL "tile" and spawn 8 threads (4 threads per core)
- 2.8x improvement

KNL: OMP Scheduling



- Because of the triangle inequality and sorted inter-centroid techniques, a given chunk of points assigned to a thread can skip computing the point-to-centroid distance calculation
- This introduces load imbalance and leads to sub-optimal performance
- Dynamic loop scheduling improves performance by 1.4x over static partitioning

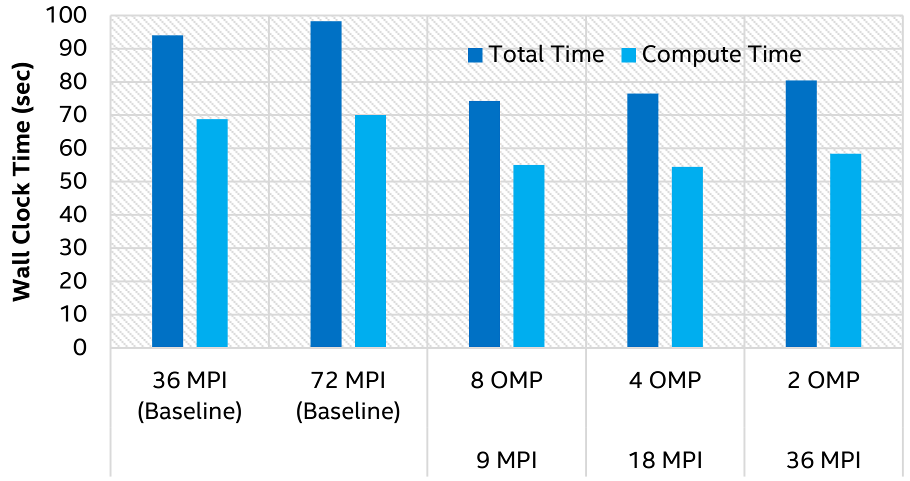
KNL: MCDRAM



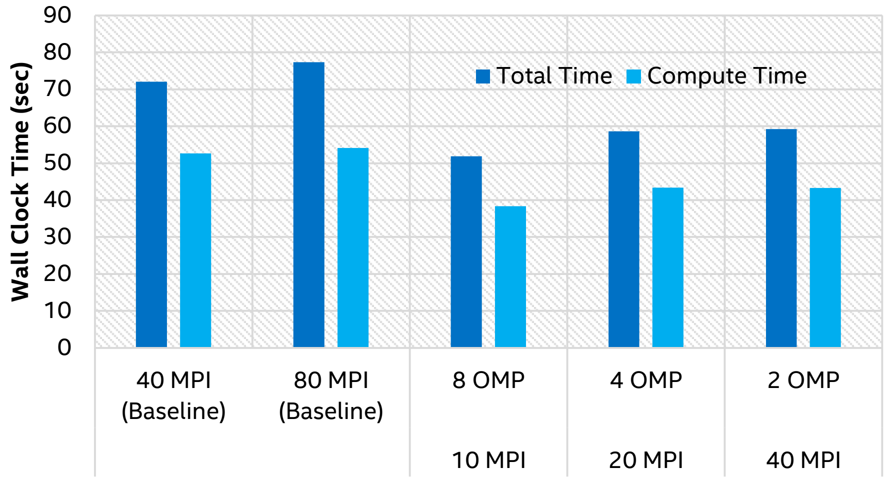
With higher volume of memory requests, MCDRAM gives 2.6x better performance

Performance Optimizations: BDW, SKX

BDW (36C/72T): Impact of HyperThreading



SKX (40C/80T): Impact of HyperThreading

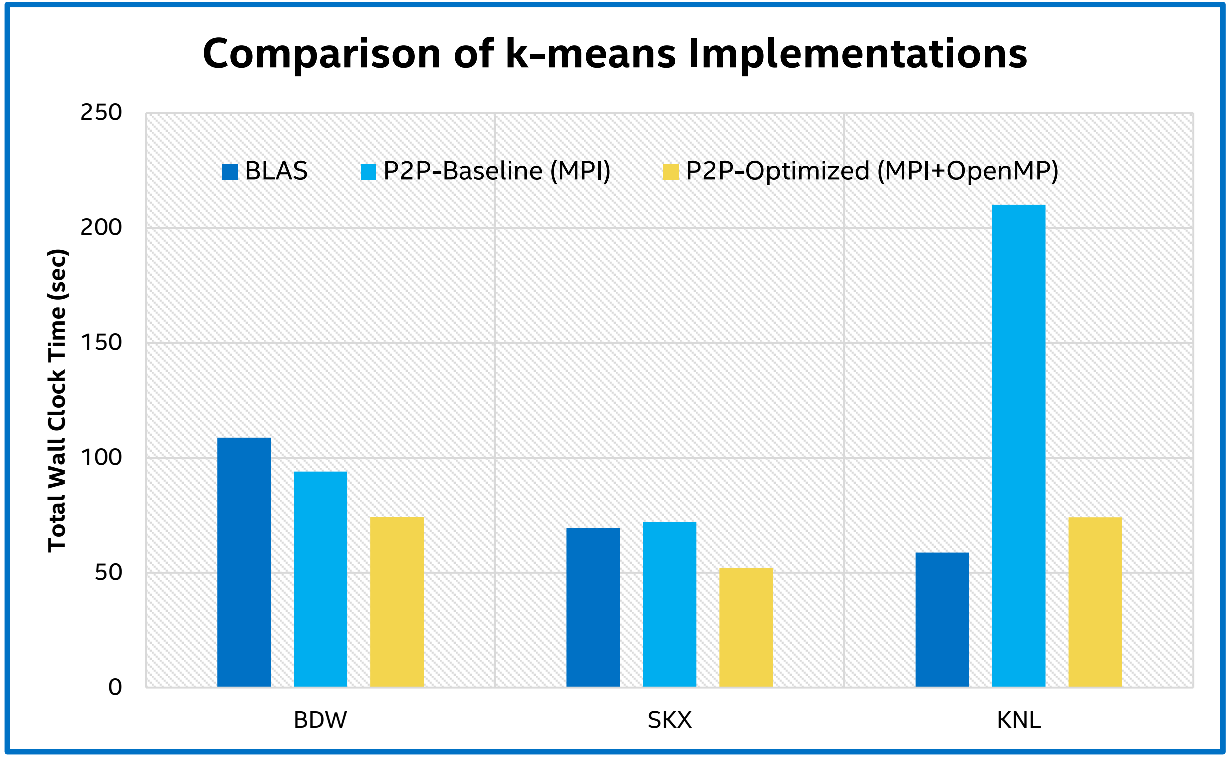


- ❑ Hybrid MPI-OpenMP implementation enables to effectively use hyper threads/logical threads
- ❑ BDW: 26% improvement with 9 MPI and 8 OMP
- ❑ SKX: 38% improvement with 10 MPI and 8 OMP

k-means as BLAS Formulation

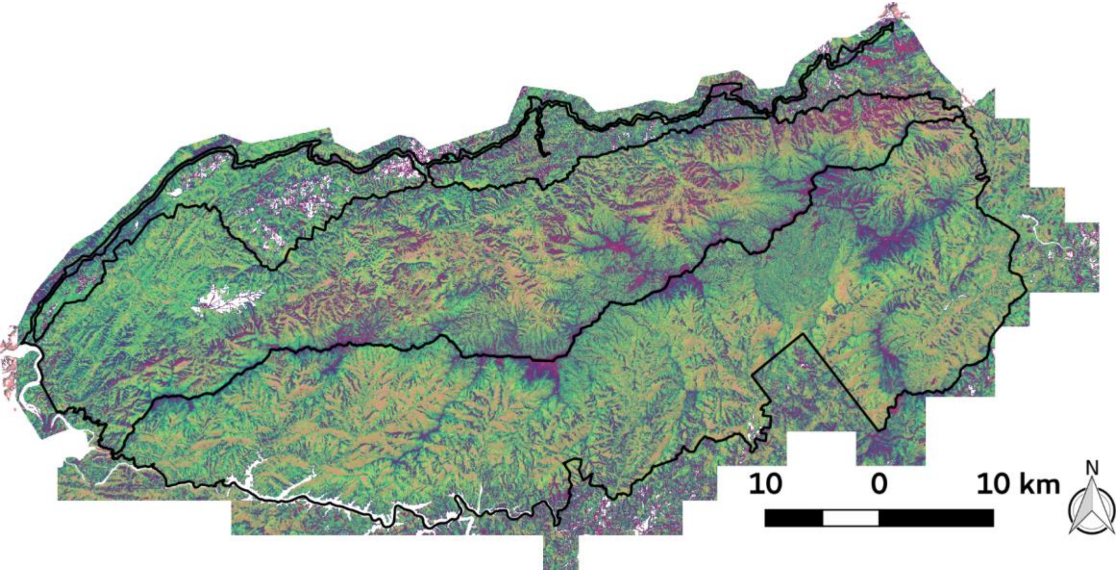
- For observation vector x_i and centroid vector z_j, the squared distance between them is $D_{ij} = ||x_i - z_j||^2$
- Binomial expansion: $D_{ij} = ||x_i||^2 + ||z_j||^2 - 2 * x_i * z_j$
- The matrix of squared distances can thus be expressed as $\mathrm{D} = \bar{x}\, 1^{\boldsymbol{T}} + 1\, \bar{z}^{\boldsymbol{T}} + 2\, X^T\, Z$, where X and Z are matrices of observations and centroids, respectively, stored in columns, and $\bar{x}$ and $\bar{z}$ are vectors of the sum of squares of the columns of X and Z, and 1 is a vector of all 1s
- The above expression for D can be calculated in terms of a level-3 BLAS operation (xGEMM), followed by two rank-one updates (xGER, a level-2 operation)
- Use Intel Math Kernel Library (MKL) to extract the best possible performance for BLAS functions

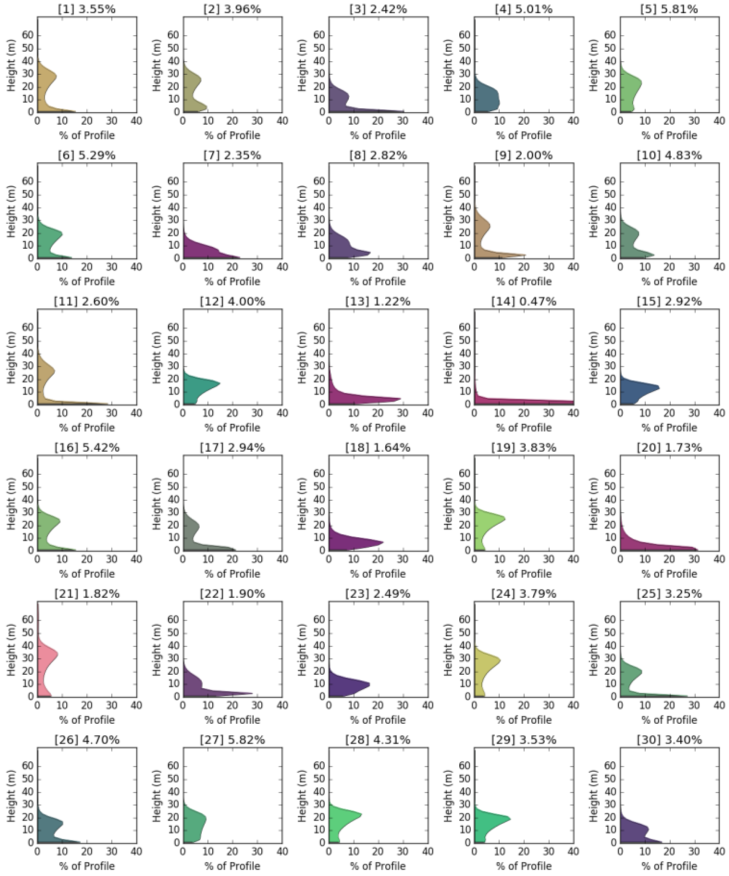
Performance Summary



- BLAS formulation provides the best performance on KNL, but slower than P2P distance calculation on BDW and SKX
- Overall performance improvements
  - KNL: 3.5x
  - BDW: 1.3x
  - SKX: 1.4x

Great Smoky Mountains National Park – Vegetation Study





Future Work

- Larger datasets
  - Multiple nodes of SKX and KNL
  - Persistent Memory/NVRAM
- De-centralized version of MPI + OpenMP
- Heuristic to switch between “traditional” distance calculation and “BLAS” formulation methods

Future Trends

- Hardware Architectures
  - Compute
    - Intel Nervana ASIC
    - Neuromorphic Computing
    - FPGA's
  - Persistent Memory
    - Intel 3D Xpoint Memory
  - Lower Precision

- Software Optimizations
  - Parallelization
  - SIMD Vectorization
  - Efficient usage of memory hierarchy
    - Caches
    - On-package high bandwidth memory
    - Persistant Memory

Legal Disclaimer

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

intel®