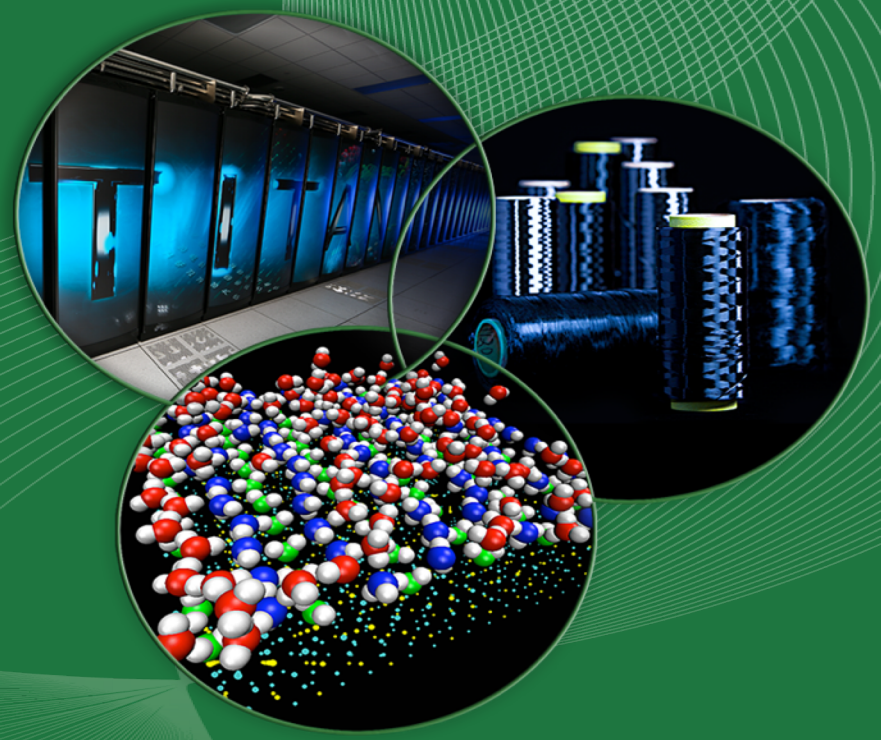
Using Compiler Directives for Performance Portability in Scientific Computing: Kernels from Molecular Simulation

Ada Sedova*,
Andreas Tillack, Arnold Tharrington

Scientific Computing, NCCS, ORNL



This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725

ORNL is managed by UT Battelle for the Department of Energy

OAK RIDGE National Laboratory

Performance Portability

- Software-development productivity is reduced when sections of high-performing programs must be frequently rewritten in low-level languages for new supercomputer architectures:
  - increased labor costs
  - code can become more error-prone
    - shortened lifetimes
    - multiple authors
    - inherent difficulty of programming close to machine-level

OAK RIDGE National Laboratory

Software Portability

- There are several types of portability
  - binary portability is the ability of the compiled code to run on a different machine
  - source portability is the ability of the source code to be compiled on a different machine and then executed
- Degree of portability (DP): $DP = C_P / C_R$, where C_P is the cost to port and C_R is the cost to rewrite the program
  - a completely portable application has an index of one, and a positive index indicates that porting is more profitable
  - **costs can include development time and personnel compensations, as well as error production, reductions in efficiency or functionality, and even less tangible costs such as worker stress or loss of resources for other projects**

OAK RIDGE National Laboratory

Performance Portability

- We can say that an application is performance portable if it is not only source-portable to a variety of HPC architectures using the Linux operating system and commonly provided compilers, but also that its performance remains in an ***acceptable range to be usable by domain scientists for competitive research***
- To avoid the ambiguity in the phrase ``acceptable range,'' Pennycook (2016) proposed the following metric for PP:

$$PP(a,p,H) = \begin{cases} \dfrac{|H|}{\sum_{i \in H} \frac{1}{e_i(a,p)}} & \text{if } a \text{ is supported } \forall i \in H \\ 0 & \text{otherwise,} \end{cases}$$

$|H|$ is the cardinality of the set H of all systems used to test the application a, p are the parameters used in a, and e_i is the efficiency of the application on each system i in H. Efficiency can be the ratio of performance of the given application to either the best-observed performance, or the peak theoretical hardware performance.

OAK RIDGE National Laboratory

- Because of such considerations, creating performance portable applications has become an important effort in scientific computing and is recognized as a significant software design goal by both the U.S. Department of Energy (DOE) and the National Science Foundation (NSF):

https://www.lanl.gov/asc/doe-coe-mtg-2017.php

NSF/Intel partnership on computer assisted programming for heterogeneous architectures (CAPA).

OAK RIDGE National Laboratory

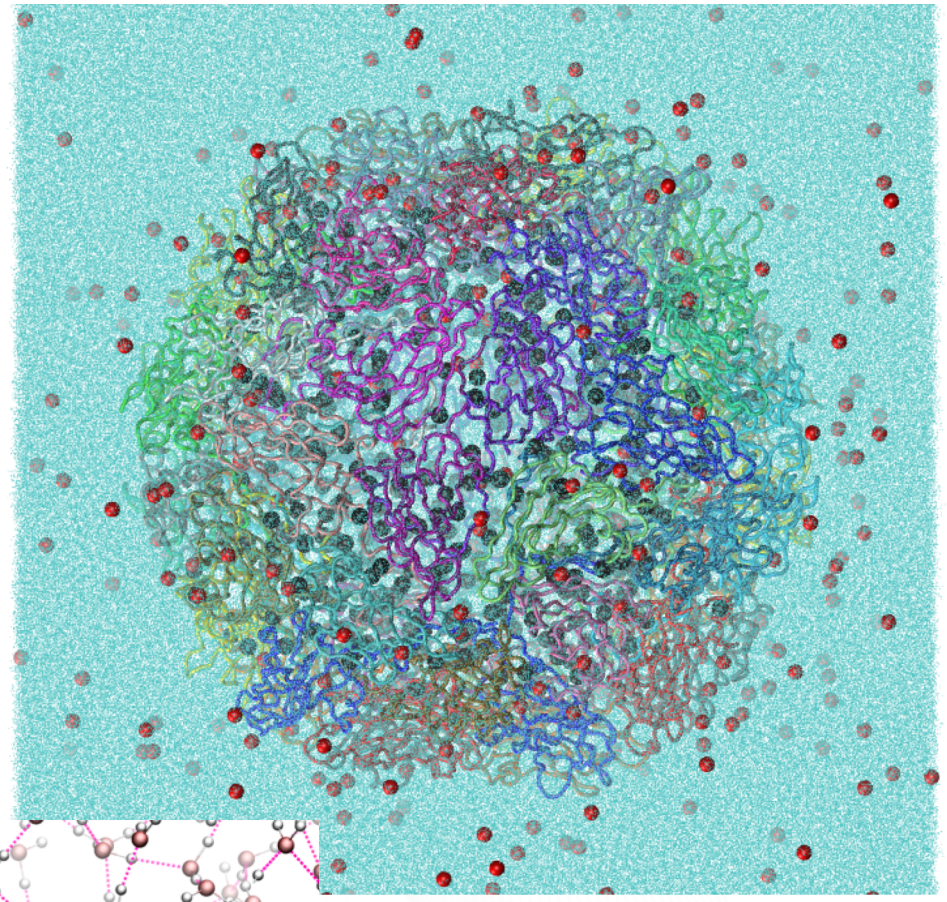
Accepted Best Practices for Portable Software Design

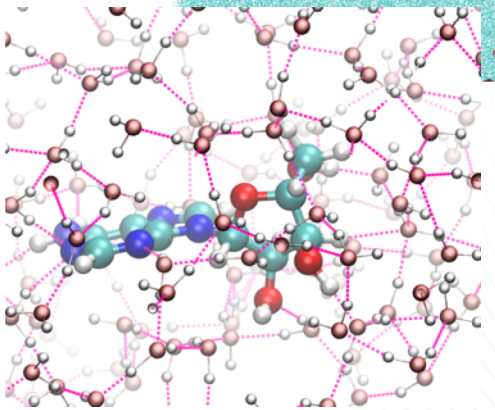
- Modular format, linear arrangement (avoid heavy nesting)
- Simplicity, clarity and ample commenting
- **Dedicated portable version**: design decisions guided by portability from initiation
- **Use of a high-level programming interface with a re-targetable back end that is standardized and supported by a number of both commercial and open-source initiatives**
  - Compiler directives for HPC parallelization?

OAK RIDGE National Laboratory

Classical Molecular Dynamics

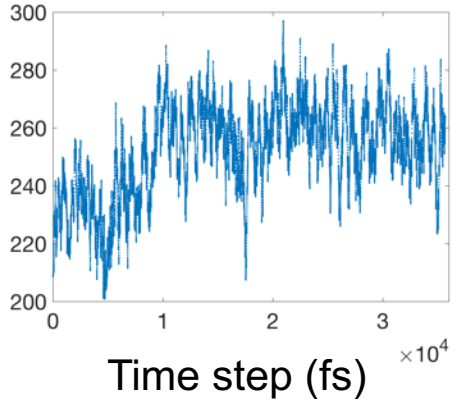
- Popular tool for a number of fields within the physical and chemical sciences
- Successfully implemented in the HPC setting by several developers
- **Extensive effort involved in porting these programs to different HPC platforms in order to meet increasingly rising standards**.
- A variety of non-portable components are employed in leadership MD programs that allow for cutting-edge performance to be obtained.
  - CUDA
  - SIMD intrinsic functions

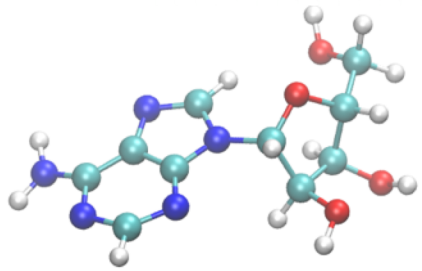




OAK RIDGE National Laboratory

- A system, represented by atomistic units, is propagated in time based on some calculated forces using a numerical integration of Newton's equations of motion.
  - **The simulation cannot proceed with the next step until the previous one is completed**
- A very small time-step is required to keep the simulation from sustaining unacceptable drifts in energy, as compared to experimental timescales that the simulation may be modeling
  - **Minimization of time per step is highly important.**
- Several open-source, highly parallel classical MD programs scale to thousands of nodes of a supercomputer and are heavily used internationally for molecular research.
  - **These programs are able to perform a time step in less than two milliseconds for systems of hundreds of thousands of atoms, or in seconds for systems of hundreds of millions of atoms**





OAK RIDGE National Laboratory

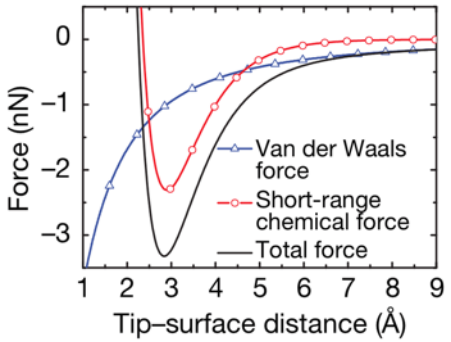
Performance Portability with OpenACC

- Testing key kernels of the basic MD algorithm using acceleration with OpenACC
  - Assess whether the resulting performance of these kernels is within an acceptable range to be used as part of HPC-based MD programs.
  - **Starting from scratch**: not restricted by non-portable decisions and legacy portions of existing applications
  - Provides tests of the performance of OpenACC on kernels that involve **non-negligible memory operations, and large memory transfers to the GPU, characteristic of many scientific applications**.
  - The kernels also represent **calculations important to other types of computational work such as classification and data analysis**.

OAK RIDGE National Laboratory

Largest Bottleneck in MD: Short-Range Non-bonded forces

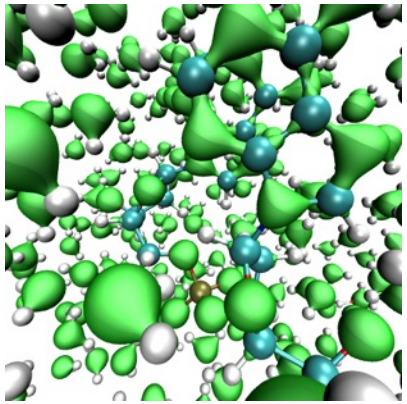
- Lennard-Jones and Coulomb intermolecular forces: pairwise distance approximation



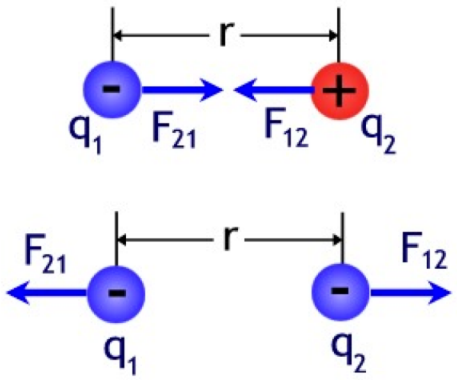
DOI: 10.13140/RG.2.2.35138.07360

$$F_{LJ}(\mathbf{r_{ij}}) = \left[12\left(\frac{\sigma_{ij}^{12}}{r_{ij}^{13}}\right) - 6\left(\frac{\sigma_{ij}^{6}}{r_{ij}^{7}}\right)\right]\frac{\mathbf{r}_{ij}}{r_{ij}},$$

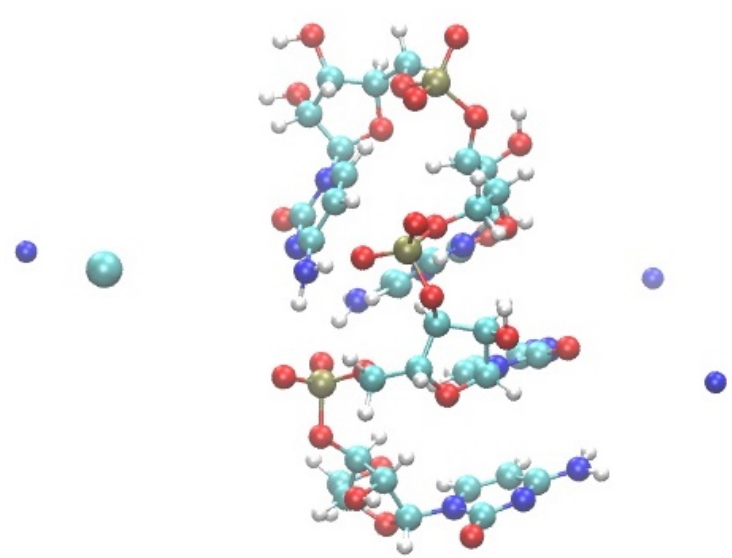
$$F_C(\mathbf{r_{ij}}) = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}^2}\frac{\mathbf{r}_{ij}}{r_{ij}}.$$



http://www.sci-news.com/othersciences/physicalchemistry/wave-like-nature-van-der-waals-forces-03717.html



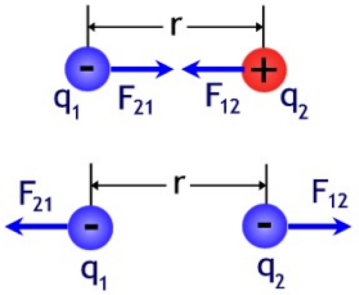
https://physics.stackexchange.com/questions/286907/what-is-the-truth-of-the-electrostatic-force



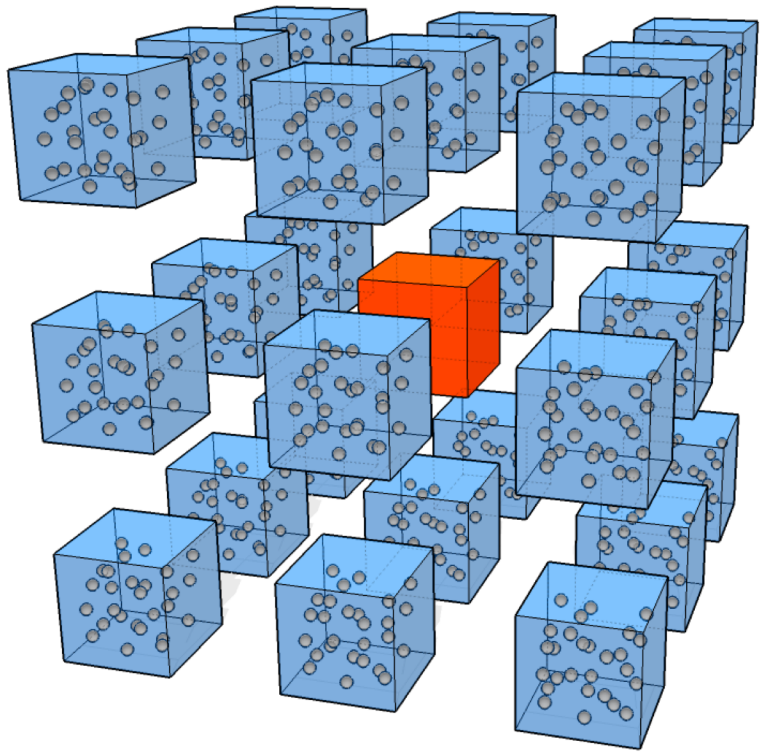
OAK RIDGE National Laboratory

Domain Decomposition

- The Lennard-Jones and short-range electrostatic forces rapidly decay to zero outside of a radius of about 10-14 angstroms.
  - Calculation can be reduced by imposing a **distance-based radial cutoff on each atom**, outside of which no interactions are considered.
  - Algorithmically, the SNF calculation usually consists of a spatial decomposition, or “domain decomposition” of the system into a **three-dimensional grid of cells**
  - If the spatial decomposition is performed so that the cells' dimensions are close to the LJ cut-off distance, then **only the interacting cell-cell pairs need to be searched for interacting atoms, for each cell**

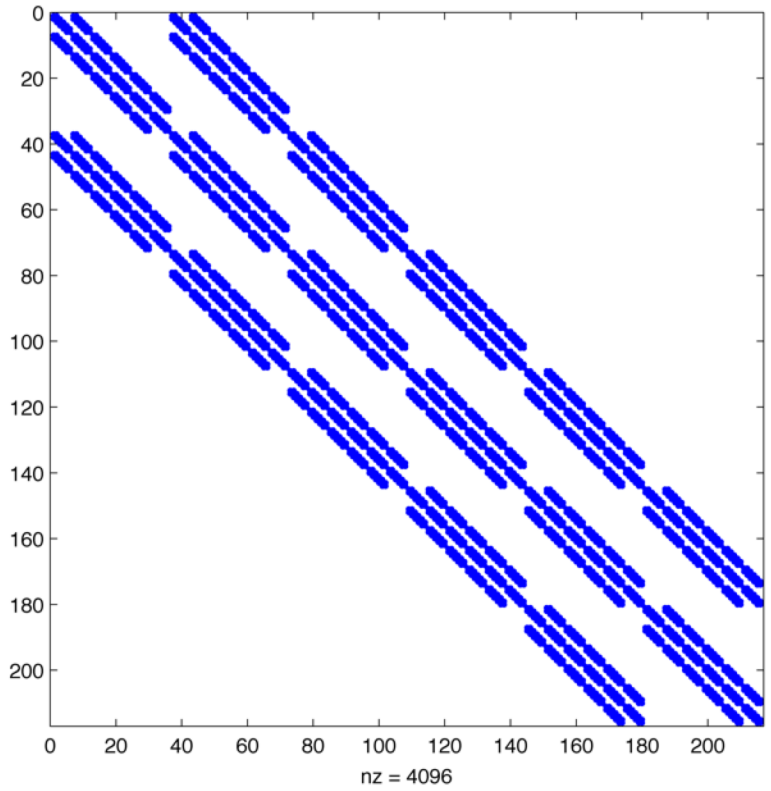


OAK RIDGE National Laboratory



A central cell and its interacting cell neighbors. In the periodic regime, all cells have 26 neighbors, and distances of all atoms within the central cell must be calculated as well, resulting in 27 cell-cell interactions that must be calculated for each cell in the grid of the domain decomposition.

Sparsity plot of the distance matrix for all cell-cell interactions in the system, with those having distances greater than the cut-off set to zero and colored white, and interacting cells colored blue. The cut-off creates a banded structure to the matrix, and reduces the number of cell-cell calculations by about 90%.



OAK RIDGE National Laboratory

Performance Goal

- Key kernels from a MD calculation whose total time should remain under 7 ms/time-step for a system under 20 M atoms, or 55 ms/time-step for a system of about 220 M atoms, after domain decomposition.
- On a single node, the job size on the node must be large enough so that the total domain decomposition would not use more that about 2000 nodes for a smaller system, and 4000 nodes for a larger system.
  - Common domain decomposition for MD programs involves computing the SNFs acting on about 15 K atoms on a single node. For around 15 K atoms, there are about 3,000 cell-cell interactions, so what we aim for is a total kernel time under 6 ms for about 3,000 cell-cell interactions, or 50 ms for about 12,000 cell-cell interactions.
  - Corresponds to an 80% efficiency score compared to NAMD, and if maintained for all architectures tested, would result in a minimum of 80% performance portability score.
  - We test whether this performance can be maintained using the *same source code*, on nodes with multi-core CPUs and on heterogeneous nodes containing a GPU.

OAK RIDGE National Laboratory

Binning module (Neighbor-list updates): bin-assign, bin-count, and bin sorting

- Bin-assign is easily parallelized by OpenACC and is very fast
  - For 500,000 atoms: 115 microseconds, 30,000 atoms: 11 microseconds using one node of Titan with the GPU.

```
#pragma acc parallel loop private(temp)
for (b = 0; b < num_atoms; b++) {
// read each atom's 3 coordinates and calculate the value of the 3-digit
// address:
 temp[0] = floor((coords[b][0]/range[0]-kbinx)*num_divx);
 temp[1] = floor((coords[b][1]/range[1]-kbiny)*num_divy);
 temp[2] = floor((coords[b][2]/range[2]-kbinz)*num_divz);
// find the 1-digit address of the atom
 bin_ids[b]= num_divy*num_divz*(temp[0])+num_divz*(temp[1])+(temp[2]);
}
```

- Bin count and sort are not easy to parallelize and can employ libraries if needed, however, due to the limited number of atoms per cell, this is also not a critical region

OAK RIDGE National Laboratory

The squared pairwise distance calculation: performance, portability, and effort

- OpenACC on the GPU and on the CPU, compared to…
- A CUDA kernel
- A method completely using routines from newly emerging batched versions of accelerator-based Basic Linear Algebra Subprograms (BLAS) libraries.
  - A pedagogical example of a solution that is not only portable, but actually requires the least amount of parallel programming experience: allows the user to perform the calculation without any knowledge of accelerator programming or even any experience with compiler directives.
  - Batched versions of BLAS standard routines are not technically part of the standard, but there is a growing need for these types of routines and they are available in many scientific libraries.
  - On the CPU, the single-batch BLAS version using MKL outperforms a naïve parallelization with openMP

OAK RIDGE National Laboratory

```
#pragma acc data copyin(A[0:3*num_batch*num_coords_n]),copyin(B[0:3*
    num_batch*num_coords_m])

pairwise_batched(A, B, C, num_coords_n, num_coords_m, num_batch);

void pairwise_batched(double A[], double B[], double C[], int ldX, int
    ldY, int num_batch){
#pragma acc data present(A), present(B), present(C)
#pragma acc kernels
#pragma acc loop independent
  for (int b=0; b<num_batch; b++)
  {
    double y[3];
    double temp;
    /* rows=dim, columns=obs. */
#pragma acc loop independent
    for (int i = 0; i < ldX; i++) {
#pragma acc loop independent
      for (int j = 0; j < ldY; j++) {
#pragma acc loop seq
        for (int k = 0; k < 3; k++)
        {
          double temp =  A[b*3*ldX + k + 3 * i] - B[b*3*ldY + k + 3 * j
    ];
          y[k] = temp * temp;
        }
        temp = y[0];
#pragma acc loop seq
        for (int k = 0; k < 2; k++)
        {
          temp += y[k + 1];
        }
        C[b*ldX*ldY + i + ldX * j] = temp;
      }
    }
  }
}
```


OAK RIDGE National Laboratory

```
template<typename T, int BS>
__global__ void
distance_kernel (T** A_data, T** B_data, T** C_data, int lda)
{
  int row_stride = lda;
  int row = blockIdx.x*BS+threadIdx.x;
  int col = blockIdx.y*BS+threadIdx.y;
  __shared__ T *A, *B, *C;
  if(threadIdx.x+threadIdx.y==0)
  {
    A = A_data[blockIdx.z];
    B = B_data[blockIdx.z];
    C = C_data[blockIdx.z];
  }
  __syncthreads();
  if ((row < lda) && (col < lda))
  {
    T elementSum = (T)0.0;
#pragma unroll
    for(int i=0; i<3; i++)
    {
      T diff = A[i*row_stride+row] - B[i*row_stride+col];
      elementSum += diff*diff;
    }
    C[col * row_stride + row] = elementSum;
  }
}
void
cuda_distance(double** A_data, double** B_data, double** C_data,
              int lda, int numBatches)
{
  const int BS = 16;
  int NB = (lda+BS-1)/BS;
  dim3 dimBlock(BS,BS);
  dim3 dimGrid(NB,NB,numBatches);
  distance_kernel<double,BS><<<dimGrid,dimBlock>>>
  ((double**)A_data, (double**)B_data, (double**)C_data, lda);
}
```

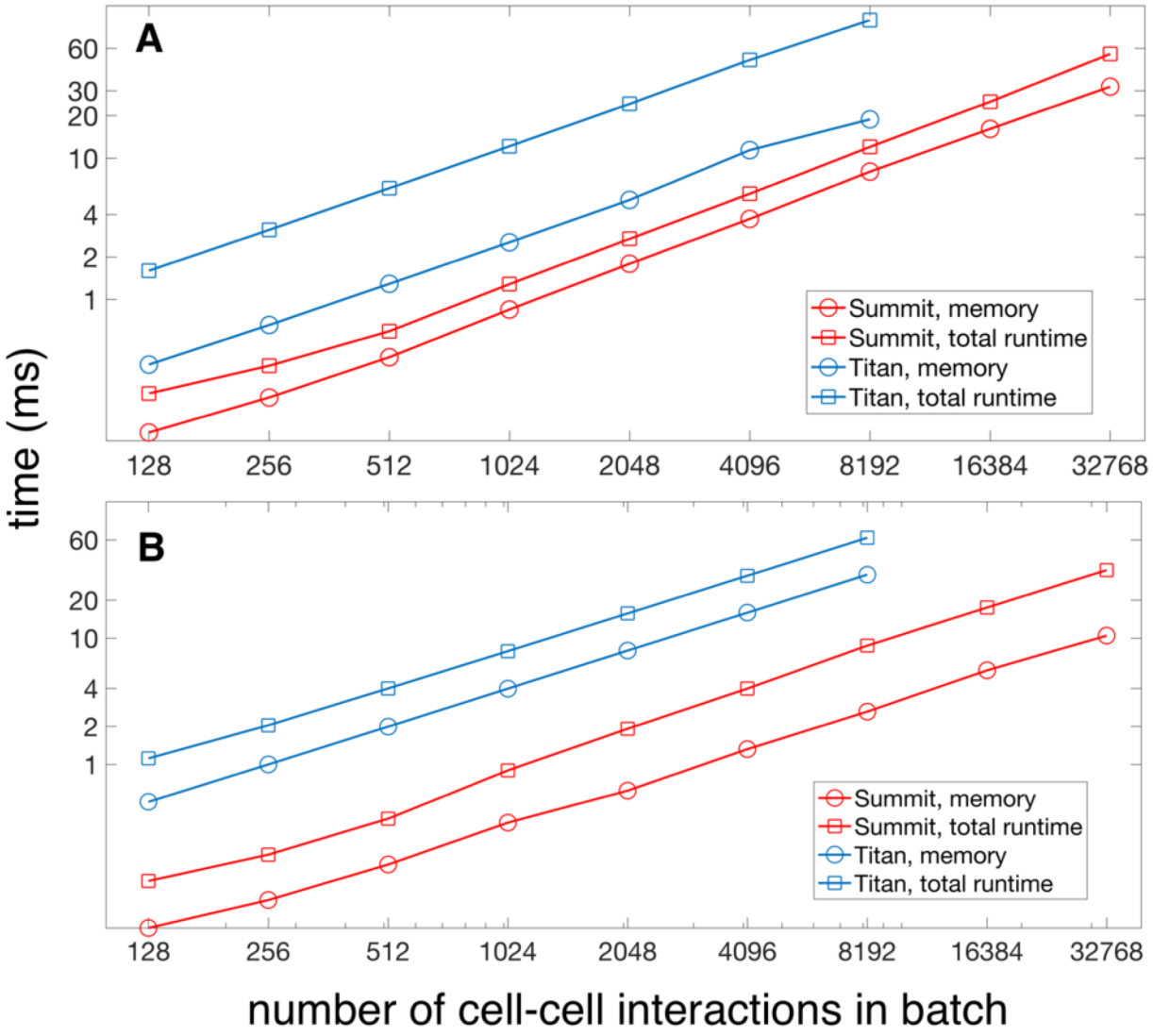

OAK RIDGE National Laboratory

Algorithm 1 Pairwise squared distance calculation using matrix operations, adapted form Li et al., 2011 [49]

load matrices $\mathbf{A}$ and $\mathbf{B}$ and allocate memory for matrix $\mathbf{C}$
$\mathbf{A}$ has dimension N by 3, $\mathbf{B}$ has dimension M by 3, and $\mathbf{C}$ has dimension N by M
note: $(\cdot)$ denotes elementwise multiplication
$\mathbf{v}_1 = (\mathbf{A} \cdot \mathbf{A})[1,1,1]^{\mathrm{T}}$
$\mathbf{v}_2 = (\mathbf{B} \cdot \mathbf{B})[1,1,1]^{\mathrm{T}}$
$\mathbf{P}_1 = [\mathbf{v}_1, \mathbf{v}_1, ..., \mathbf{v}_1]$ (dimension N by M)
$\mathbf{P}_2 = [\mathbf{v}_2, \mathbf{v}_2, ..., \mathbf{v}_2]^{\mathrm{T}}$ (dimension N by M)
$\mathbf{P}_3 = \mathbf{A}\mathbf{B}^T$(dimension N by M)
$\mathbf{D}^2 = (\mathbf{P}_1 + \mathbf{P}_2 - 2\mathbf{P}_3)$, where $\mathbf{D}^2$ is the matrix of squared distances
pairwise distance matrix can be recovered from $\mathbf{D}^2$ by element-wise square-root

OAK RIDGE National Laboratory

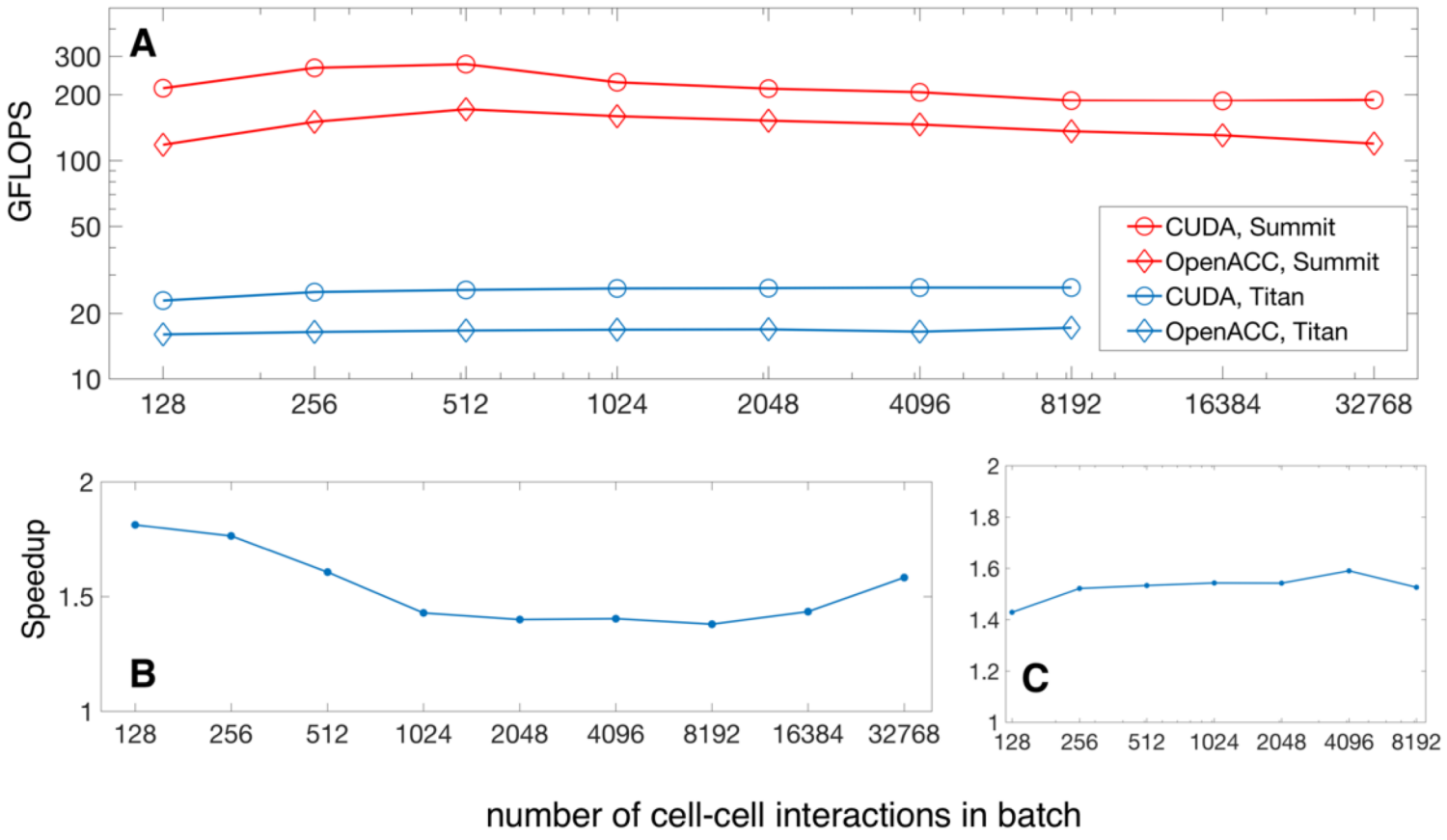
Performance on the GPU



- On both Titan and Summit, a reasonable number of batches could be processed in under 10 ms, and on Summit, all cell-cell interactions for a system the size of a small protein (about 6000 batches) could be processed on a single GPU in under 10 ms.
  - These results are within the acceptable range we determined for an MD step.

Comparison of performance (time in ms), for CUDA and OpenACC versions of the GPU-based all-pairwise squared distances calculation on OLCF Titan (K20X) and Summit (V100), over increasing batch sizes. A: Using OpenACC distance kernel. B: Using CUDA distance kernel.

OAK RIDGE National Laboratory

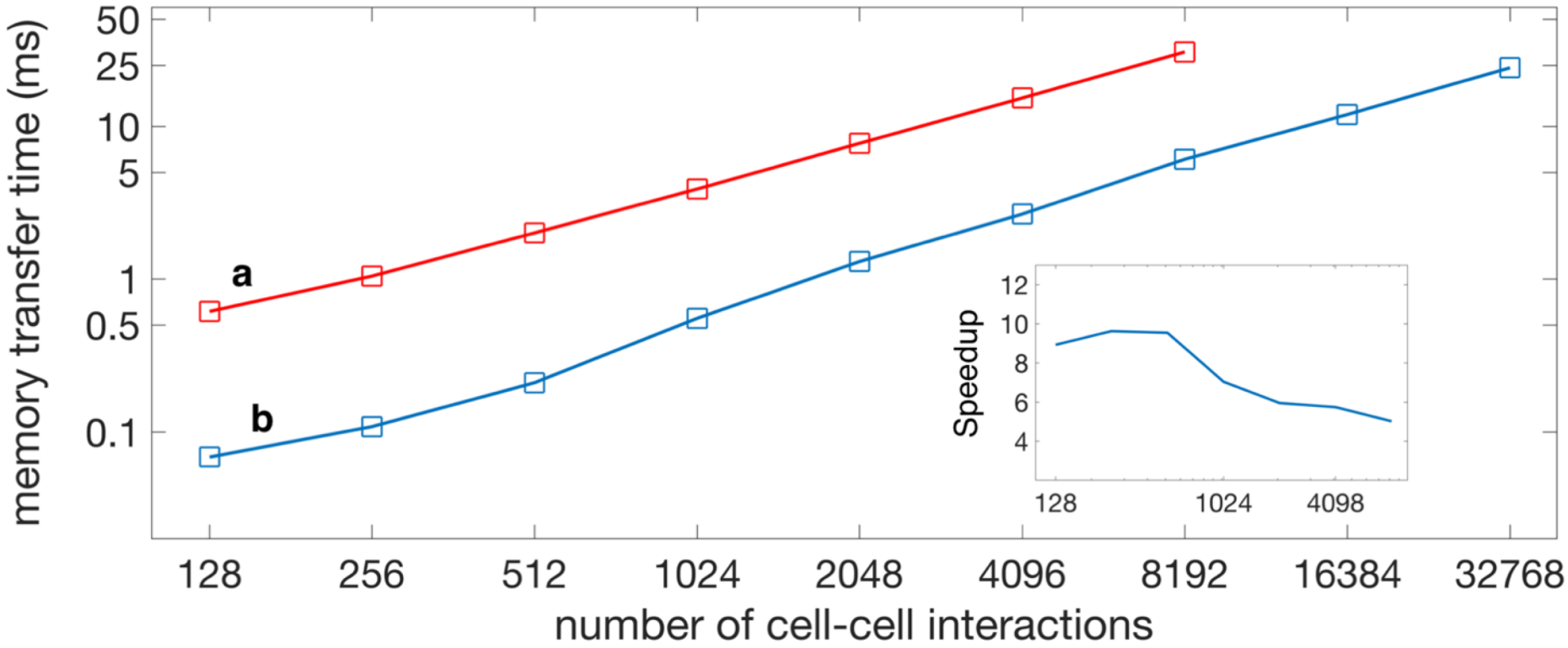


A. Comparison of performance by GFLOPS for CUDA and OpenACC versions of the GPU-based all-pairwise squared distances calculation on OLCF Titan (K20X) and Summit (V100), over increasing batch size. **B:** Speedup (✖) of CUDA kernel over OpenACC kernel distance kernel on Summit, for total runtime and memory transfer time. **C:** Speedup of CUDA kernel over OpenACC kernel distance kernel on Titan.

The upper limit on the time-per-step greatly constrains the amount of batches that can be offloaded to a single node, resulting in the use of only a small percent of the peak FLOPs available on the GPU. With no such constraint, it would be possible to perform significantly more pairwise distance calculations per node in a relatively rapid amount of time based on how much the two tested GPUs' global memories can hold.

OAK RIDGE National Laboratory

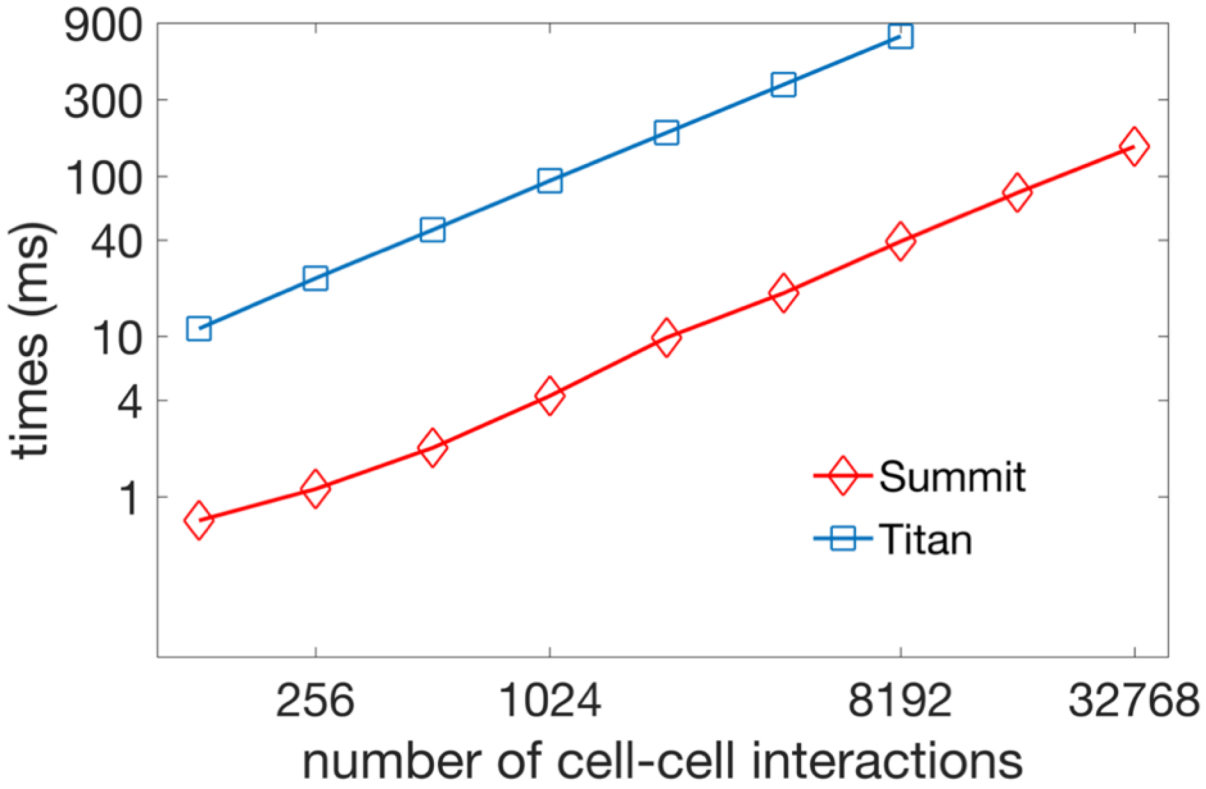
Memory transfer time to GPU

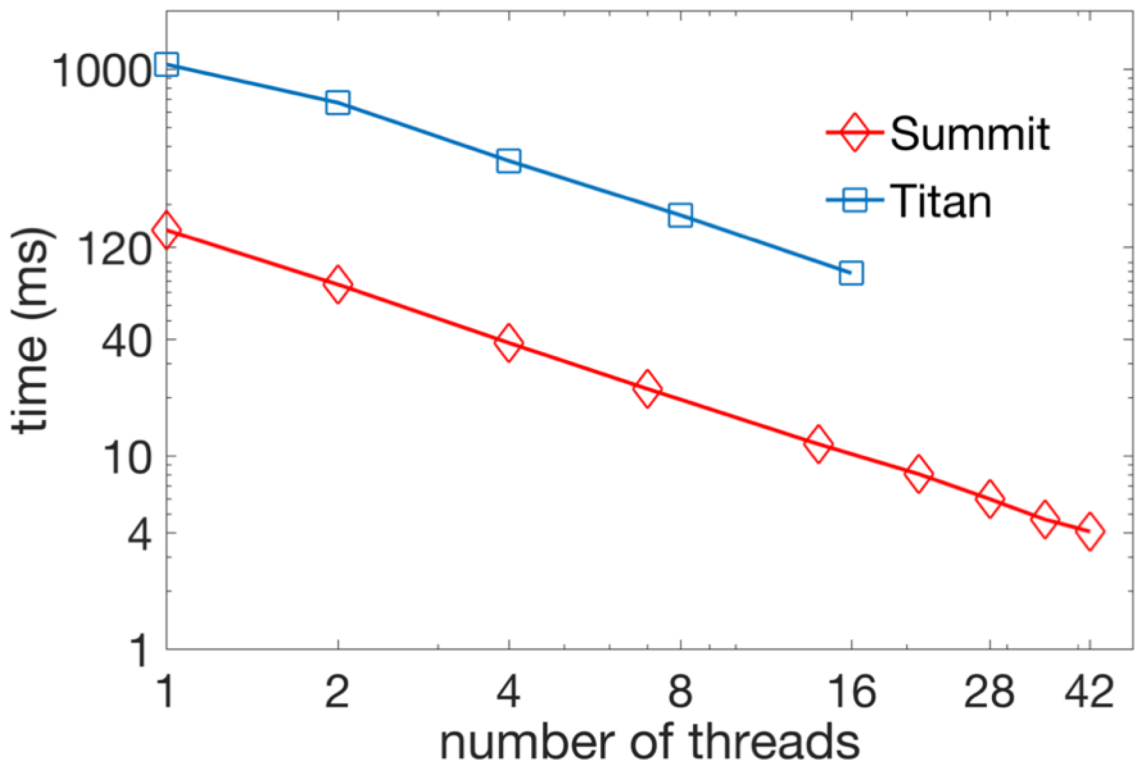


Comparison of memory transfer time for CUDA versions (and BLAS version) of the GPU-based all-pairwise squared distances calculation on a) OLCF Titan (K20X) and b) Summit (V100), for different batch sizes. Inset: Speedup for Summit versus Titan

OAK RIDGE National Laboratory

Performance on the CPU





OAK RIDGE National Laboratory

Performance on the CPU

- For smaller batch sizes, times can be within an acceptable range for Summit, but not Titan.
- The small batch size limit reduces the number of cell-cell interactions to those in an equivalent system of about 20,000 atoms.
- Therefore, we see that performance of OpenACC, even on new supercomputer cores, is barely within the lowest limit for performance portability for the distance kernel.

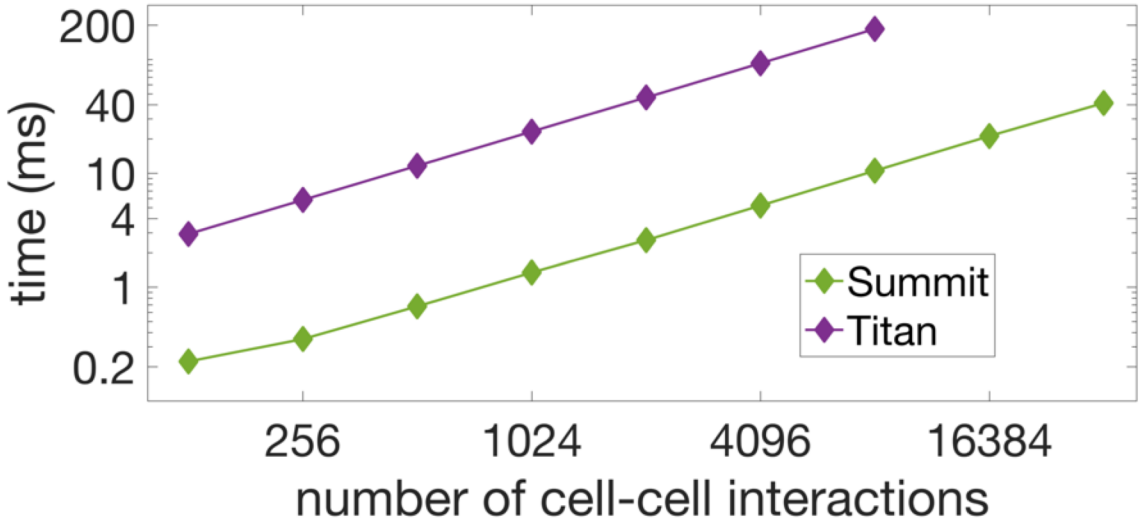
OAK RIDGE National Laboratory

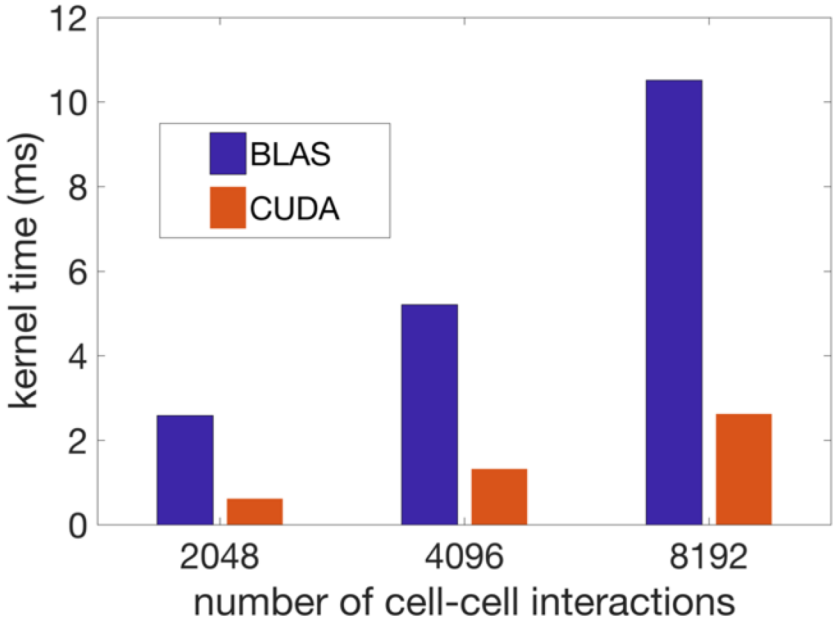
- To get maximum performance on the CPU you must use threading, alignment, and vectorization
- OpenACC has no functionality for intrinsic-function level specification. It also has no option for treating thread affinity.
- OpenACC seems to be less useful for creating truly performant CPU-based kernels than GPU version, for kernels like the distance calculation.

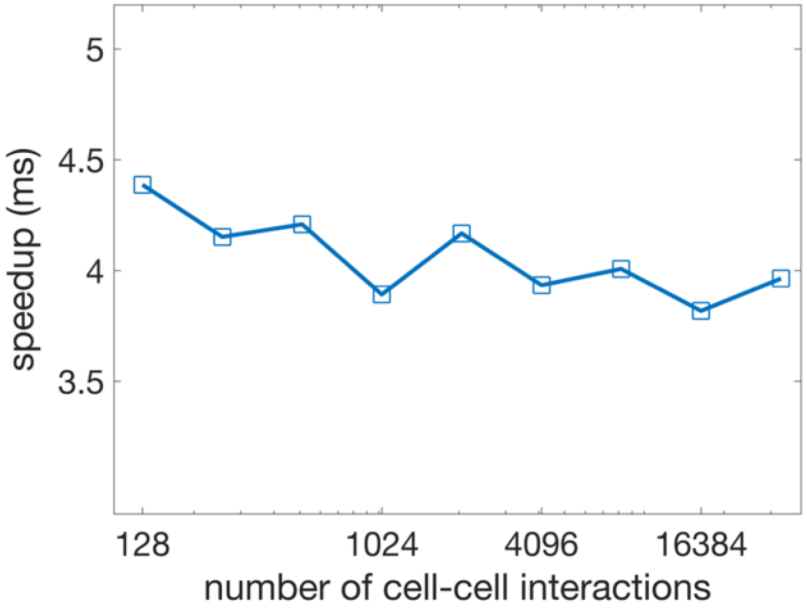
OAK RIDGE National Laboratory

Batched BLAS version

Comparison of performance of BLAS version, all-pairwise squared distances calculation, on the GPU, using OLCF Titan (K20X) versus Summit (V100).







Left: Comparison of performance (time, ms) of BLAS versus CUDA version of all-pairwise squared distances calculation, using one GPU on Summit. Right: Speedup of CUDA version on Summit vs. cuBLAS-batched version on Summit.

OAK RIDGE National Laboratory

- The element-wise multiplication is available in MKL as a Hadamard product, but not in cuBLAS, thus lines 4 and 5 of the algorithm were performed on the CPU. Because of this, we found that on the GPU, this algorithm cannot be performed completely with cuBLAS functions.
- Performance of the BLAS version is quite poor, but better than the OpenACC-threaded CPU version.
- Despite optimized BLAS routines on the GPU provided by NVIDIA, the memory operations swamp the performance in comparison to the CUDA-C and the GPU-based OpenACC versions.

OAK RIDGE National Laboratory

Amount of programming effort

- Some papers report that CUDA requires more effort than OpenACC, even for workers familiar with both APIs.
- However, different compilers each may implement a particular directive instruction differently, and variable performance may require alternate constructs to be used to parallelize a particular section of code, leading to some level of trial and error in each port.
  - This lack of a defined outcome increases potentials for performance portability, as there are more possibilities that optimal performance will be obtained by using different constructs and different compilers, but can be frustrating for the user, and increased experience may not increase the ease of this process.
  - Therefore, the use of OpenACC may not require significantly less total worker effort than use of CUDA-C for small kernels.
- On the other hand, the amount effort required for OpenACC parallelization is not large, and the result is far more portable than CUDA-C after the first implementation has been created. It is also possible that the effort required for OpenACC is less than for some alternative portable solutions, such as OpenCL.

OAK RIDGE National Laboratory

Conclusions

- For calculations representing bottleneck regions in MD, using OpenACC, performance portability can be achived.
- We found that while performance on the GPU was closer to the performance of CUDA kernels, on the CPU, performance of threaded kernels was much lower, and on older CPUs such as the AMD Bulldozers, would not provide acceptable performance. However, on the Power 9 processors, CPU performance remained within the low range of acceptability for smaller job sizes.
- Future work can compare the performance of these kernels when using OpenMP both on the CPU and the GPU. It is possible that the need for some amount SIMD-level instructions could be required for better performance on the CPU, and can also be tested in future work with OpenMP SIMD constructs.

OAK RIDGE National Laboratory

Acknowledgements

Thank you to:

OLCF, ORNL

Scientific Computing group, NCCS, ORNL

CMB, ORNL,

Oscar Hernandez

Thank you to the WACCPD Workshop Organizers

OAK RIDGE National Laboratory