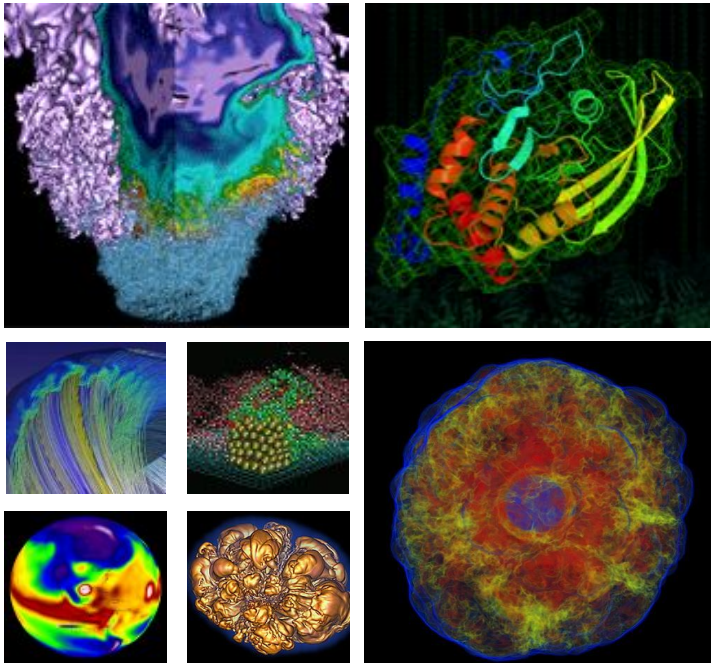
Profiling MatVec compute hierarchy



Neil Mehta, Rahul Gayatri, Yasaman Ghadar, and Jack Deslippe

NERSC

IXPUG 2020 - 13th October 2020

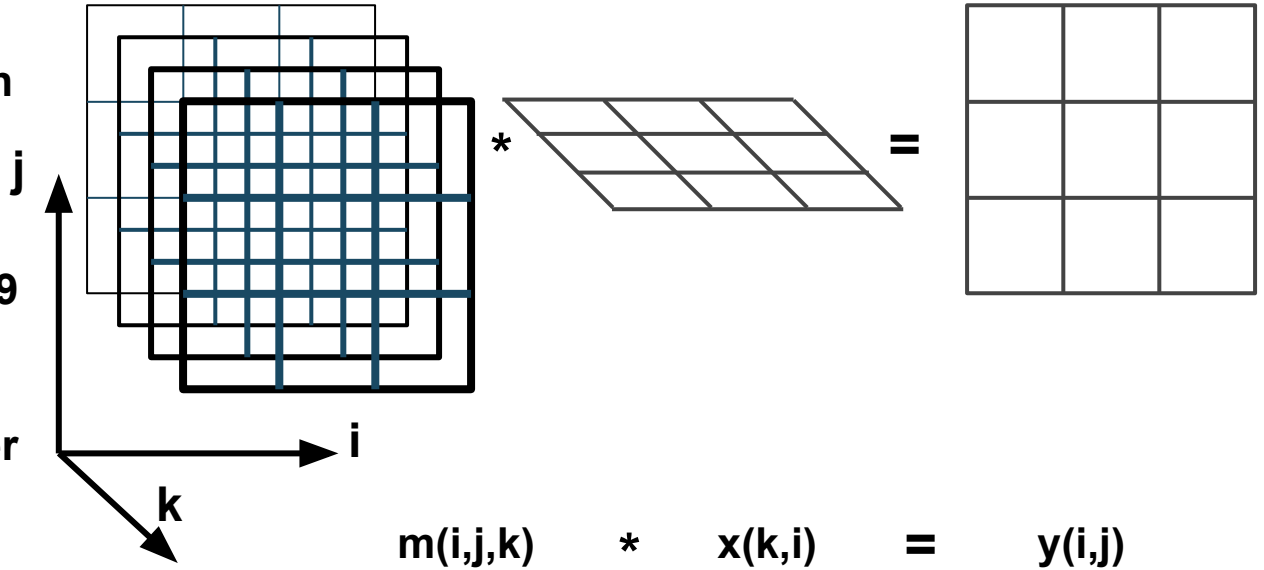
U.S. DEPARTMENT OF ENERGY | Office of Science

Argonne NATIONAL LABORATORY

BERKELEY LAB

Introduction to MatVec

- Matrix vector multiplication benchmark to measure performance portability
- Profile benchmark on NVIDIA V100 and Intel Gen9 for OpenMP 4.5 implementation
- Intel® DPC++/C++ Compiler (202008) on Intel Gen9
- clang/LLVM 11.0 on V100
- Problem size is 1000 x 1000 x 1000
- Code has 3 Gb memory footprint



m(i,j,k) * x(k,i) = y(i,j)

m(j,k) * x(k) = y(j)

Compute hierarchy in MatVec

- Algorithm divided in three compute hierarchies
- We divide work using teams distribute at first level
- At second level we use omp for
- At third level we use omp simd construct

```
#pragma omp target enter data map(to : m, x, y)
#pragma omp target enter data map(to:m.dptr [0:m.size],x.dptr[0:x.size],y.
    dptr[0:y.size])
#pragma omp target teams distribute
for (int i = 0; i < N; ++i)
{
#pragma omp parallel
  {
    team_matvec(i, m, x, y.subArray(i));
  }
}
```

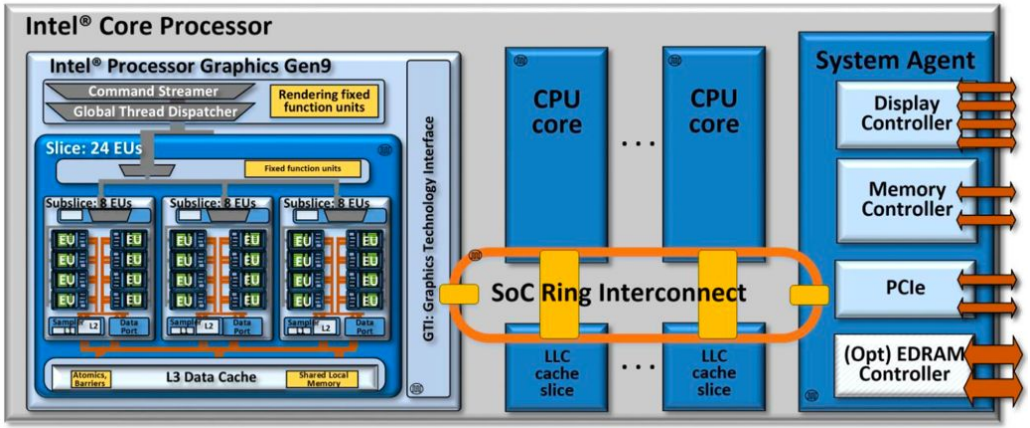
```
void
team_matvec(int i, ARRAY3D& m, ARRAY2D& x, DataType* y)
{
#pragma omp for
  for (int j = 0; j < N; ++j) {
    y[j] += vector_dot(m.subArray(i, j), x.subArray(i));
  }
}
```

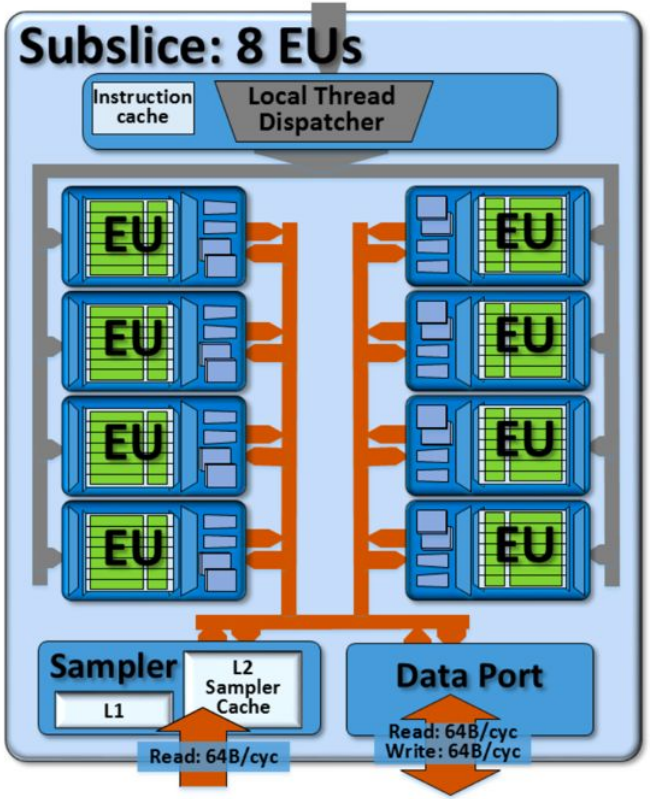
```
int
vector_dot(DataType* m, DataType* x)
{
  int result = 0;
#pragma omp simd
  for (int k = 0; k < N; ++k)
  {
    result += m[k] * x[k];
  }
  return result;
}
```

GPU architecture of NVIDIA V100

- Each SM launches maximum of 32 thread blocks
- Total of 5120 FP32 cores per GPU

GPU architecture of Intel Gen9





- Integrated GPU architecture
- 3 GPU slices, each having 3 sub-slices
- Each subslice has 8 execution unit (EU)
- Each EU has 7 thread
- Total of 504 threads per GPU

Time to solution data

NVIDIA V100

Metric	Baseline (1 thread)	With omp for	With omp simd
Kernel time (s)	11.00	2.17	2.16
Total time (s)	13.10	4.15	4.16

Intel Gen9

Metric	Baseline (1 thread)	With omp for	With omp simd
Kernel time (s)	275.43	20.36	20.41
Total time (s)	275.94	20.87	20.91

- **Currently, simd construct is not functional on either GPU architectures**

Work mapping hierarchy

NVIDIA V100

grid_size: 1000 x 1 x 1
block_size: 128 x 1 x 1

Intel Gen9

work_size_global: 32 x 1000 x 1
work_size_local: 32 x 1 x 1

```
#pragma omp target teams distribute
for (int i = 0; i < N; ++i)
{
#pragma omp parallel
  {
    team_matvec(i, m, x, y.subArray(i));
```

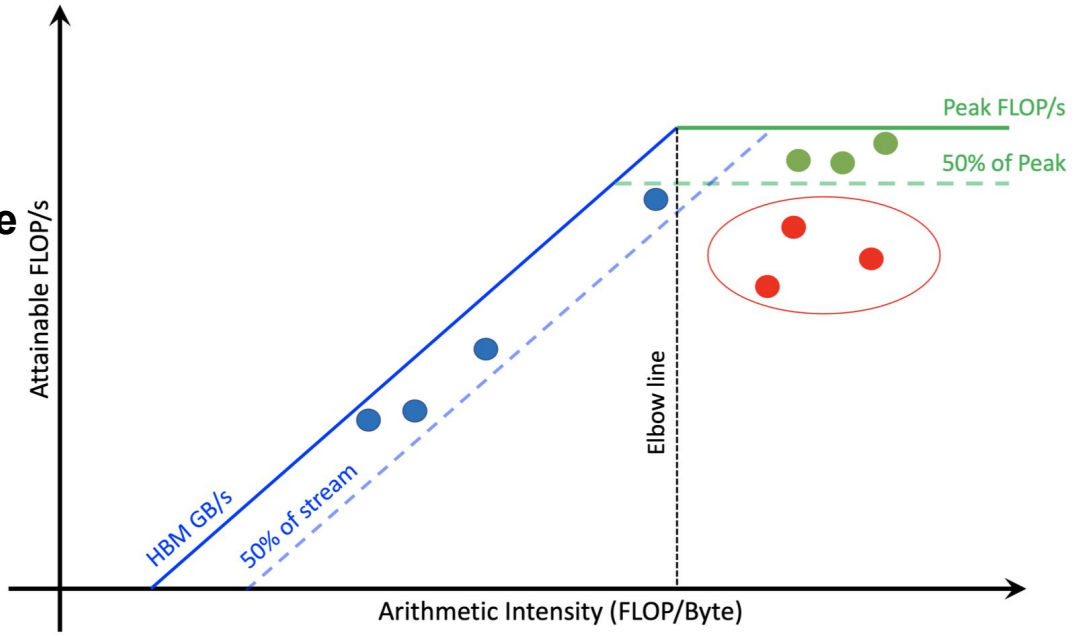
- Mapped at a level of streaming multiprocessor (SM) on NVIDIA V100
- Mapped at a level of subslices on Intel Gen9 (?)

```
team_matvec(int i, ARRAY3D& m, ARRAY2D& x, DataType* y)
{
#pragma omp for
  for (int j = 0; j < N; ++j) {
    y[j] += vector_dot(m.subArray(i, j), x.subArray(i));
```

- Mapped at a level of thread blocks on NVIDIA V100, launched in a block of 128 threads
- Mapped at a level of execution unit (EU) on Intel Gen9, launched in a block of 32 threads (?)

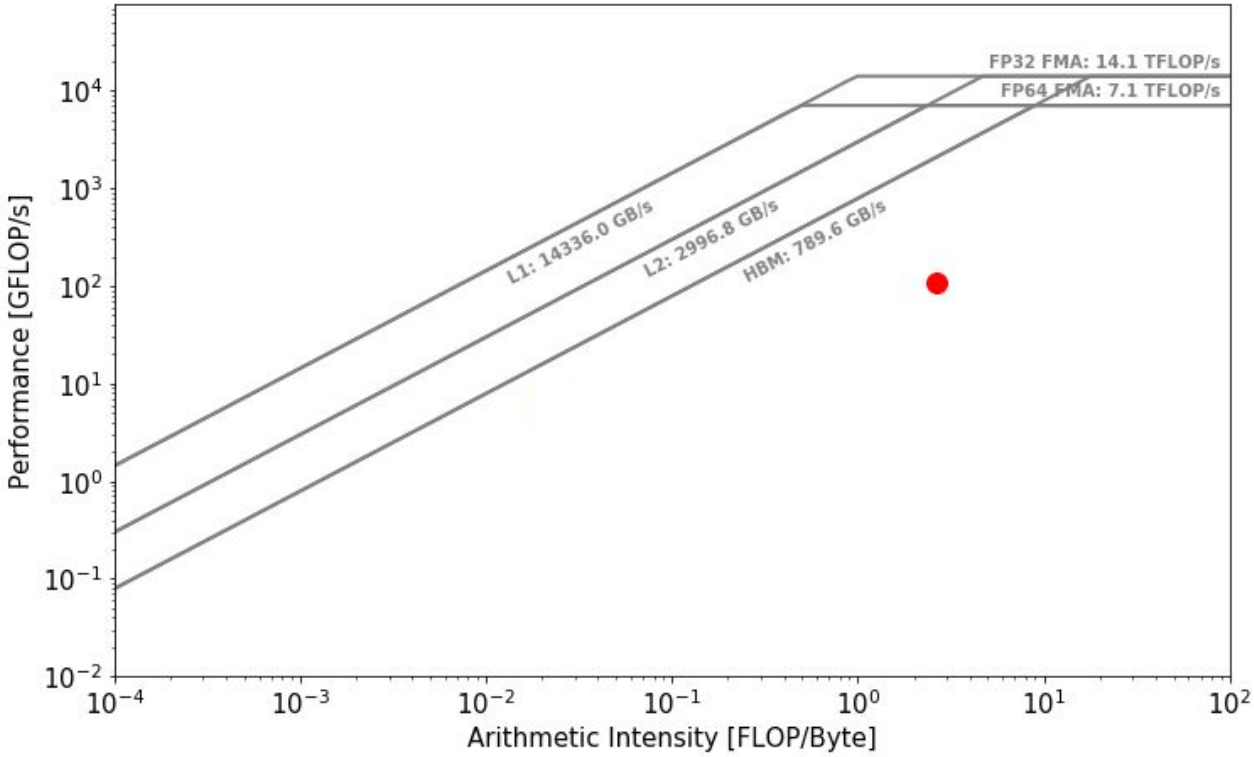
Understanding Roofline model

- Region to the left of 'elbow line' represents memory bound
- Region to the right represents compute bound
- Kernels shown in blue and green are close to peak memory bandwidth and compute throughput, representing bounded performance
- Red kernels are neither compute or memory bound and show potential for greater optimization
- Ideally kernel should shift upwards and rightwards



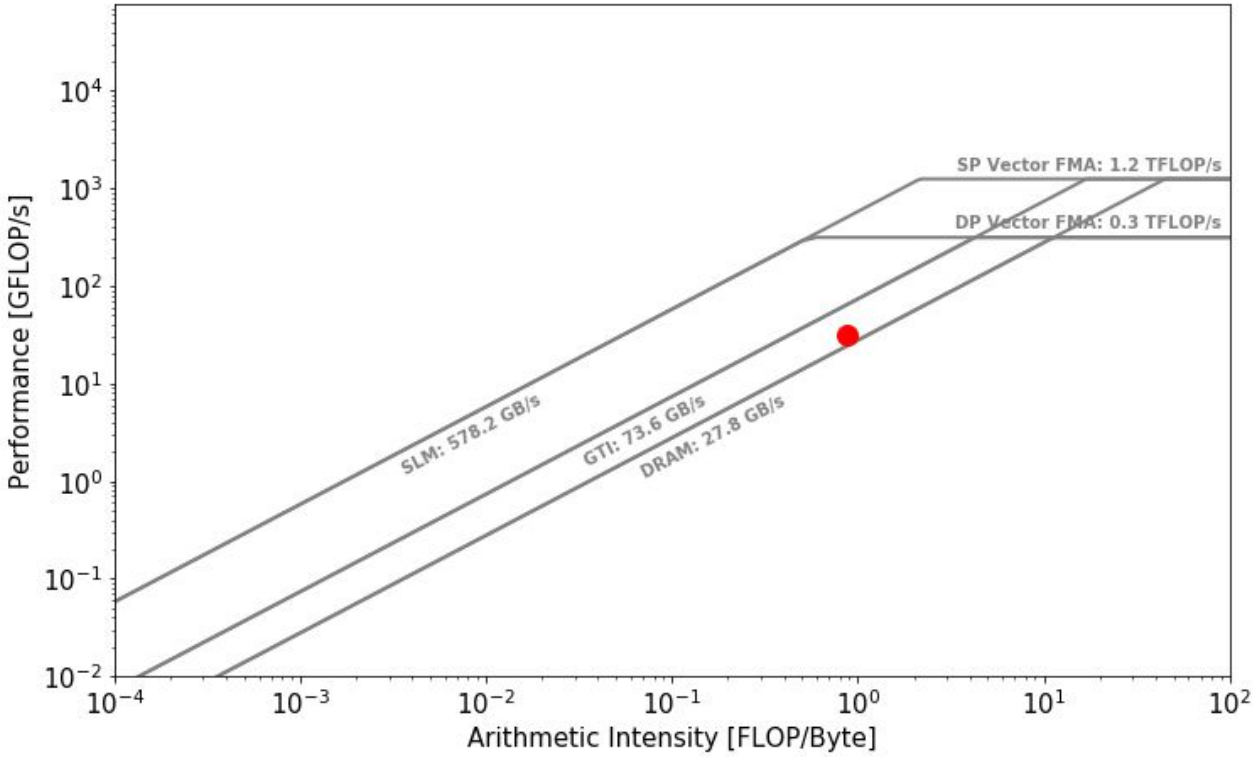
Roofline from NVIDIA V100

- Kernel is placed in memory bound region but not bound by memory bandwidth
- Kernel is closer to the elbow line and not within 50% of peak compute throughput or memory bandwidth
- Greater scope for improvement with better optimizations



Roofline from Intel Gen9

- Kernel is close to the DRAM bandwidth line, indicating memory bound
- Theoretically, roofline cannot cross bandwidth line
- Kernel crossing DRAM bandwidth indicates eDRAM memory usage
- Suggests performance capped by memory bandwidth and not compute capability



Summary

- Profiles indicate expected difference in the performance between NVIDIA V100 and Intel Gen9 as threads are approximately 10 times higher on NVIDIA V100
- Kernels on both GPUs are memory bound
- Benchmark code is bounded by DRAM bandwidth on Intel Gen9 but has not on NVIDIA
- Compiler maturity plays important role in code performance
- Need for further development in work mapping using simd

Benchmark code available at https://github.com/rgayatri23/matvec

Joint Laboratory for System Evaluation (JLSE)
https://jlse.anl.gov

We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory.

Thank You

Backup slides

Variadic versus non-variadic arrays

```
define ARRAY2D ArrayMD<int, 2>
define ARRAY3D ArrayMD<int, 3>
RRAY2D y(N, N);
RRAY2D x(N, N);
RRAY3D m(N, N, N);
```

```
#define ARRAY2D Array2D<int>
#define ARRAY3D Array3D<int>
ARRAY2D y(N, N);
ARRAY2D x(N, N);
ARRAY3D m(N, N, N);
```

- A single class to create multidimensional arrays for every dimension and data type
- Class is templated over the number of dimensions
- Variadic template pack expansion is used to calculate the offset in each dimension

- An array class is templated but only per data type
- Requires duplication of templated class for each multi-dimensional array class

Profiling data - NVIDIA V100

Metric	Baseline	With omp for	With omp simd
Kernel time (s)	10.70	1.82	1.81
Total time (s)	11.85	2.99	2.96

Metric	Variadic	Non-variadic
Kernel time (s)	1.81	7.20
Total time (s)	2.96	8.75

grid_size: 1000 x 1 x 1
block_size: 128 x 1 x 1

Profiling data - NVIDIA V100

Metric	Baseline	With omp for	With omp simd
Kernel time (s)	11.00	2.17	2.16
Total time (s)	13.10	4.15	4.16

Metric	Variadic	Non-variadic
Kernel time (s)	2.16	3.08
Total time (s)	4.16	5.81

Profiling data - Intel Gen9

Metric	Baseline	With omp for	With omp simd
Kernel time (s)	275.43	20.36	20.41
Total time (s)	275.94	20.87	20.91

Metric	Variadic	Non-variadic
Kernel time (s)	20.41	21.36
Total time (s)	20.91	22.37