OAK RIDGE National Laboratory

Frontier/Crusher Node Performance

Tom Papatheodore

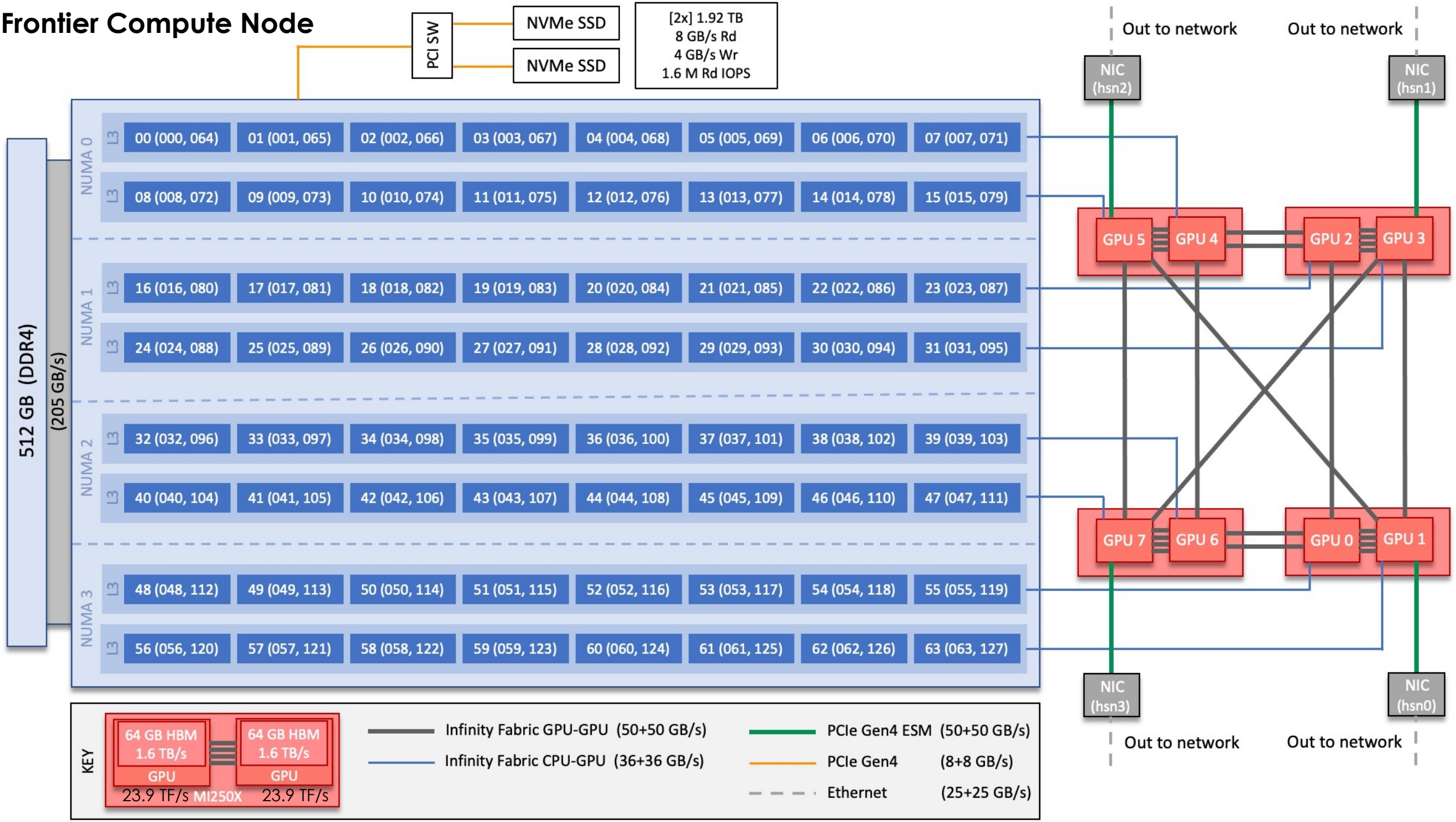
HPC Engineer
System Acceptance & User Environment Group
Oak Ridge Leadership Computing Facility (OLCF)
Oak Ridge National Laboratory (ORNL)

Frontier Training Workshop – February 16, 2023

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Frontier Compute Node



Build and run environment

Unless specified otherwise, the results in these slides were obtained using the following modules on Frontier:

```
$ module -t list
craype-x86-trento
libfabric/1.15.2.0
craype-network-ofi
perftools-base/22.12.0
xpmem/2.5.2-2.4_3.20__gd0f7936.shasta
cray-pmi/6.1.8
cce/15.0.0
craype/2.7.19
cray-dsmml/0.2.2
cray-mpich/8.1.23
cray-libsci/22.12.1.1
PrgEnv-cray/8.3.3
DefApps/default
rocm/5.3.0
craype-accel-amd-gfx90a
```

CPU DRAM Bandwidth

CPU Stream Test

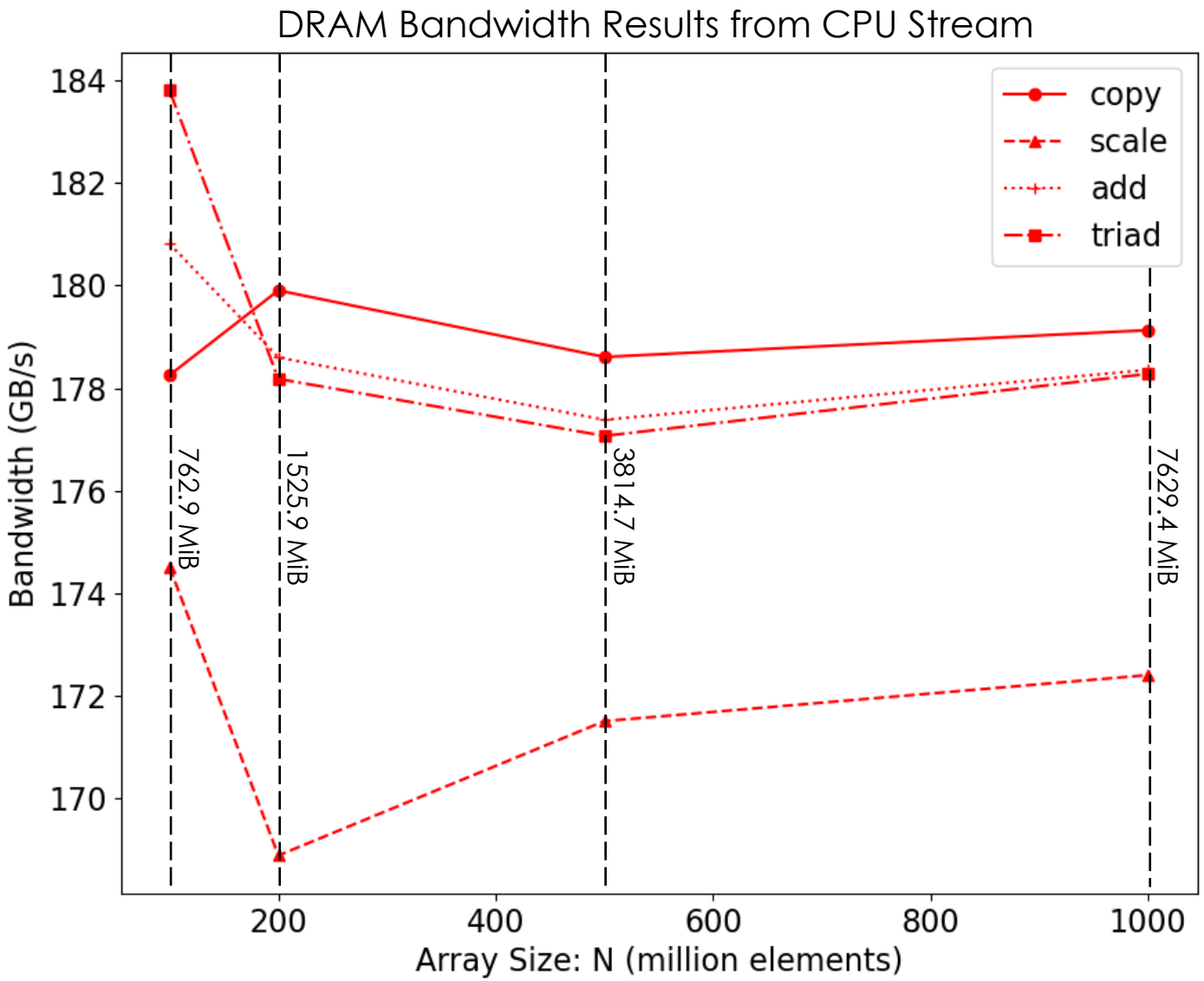
Copy	`A[i] = B[i]`
Scale	`A[i] = n*B[i]`
Add	`A[i] = B[i] + C[i]`
Triad	`A[i] = B[i] + n*C[i]`

- Peak DRAM bandwidth: 205 GB/s
- Observed DRAM bandwidth: 82-90% of peak

NOTES:

```
OMP_NUM_THREADS='8'

Each kernel will be executed 35 times.
The *best* time for each kernel (excluding
the first iteration) will be used to compute
the reported bandwidth.
```



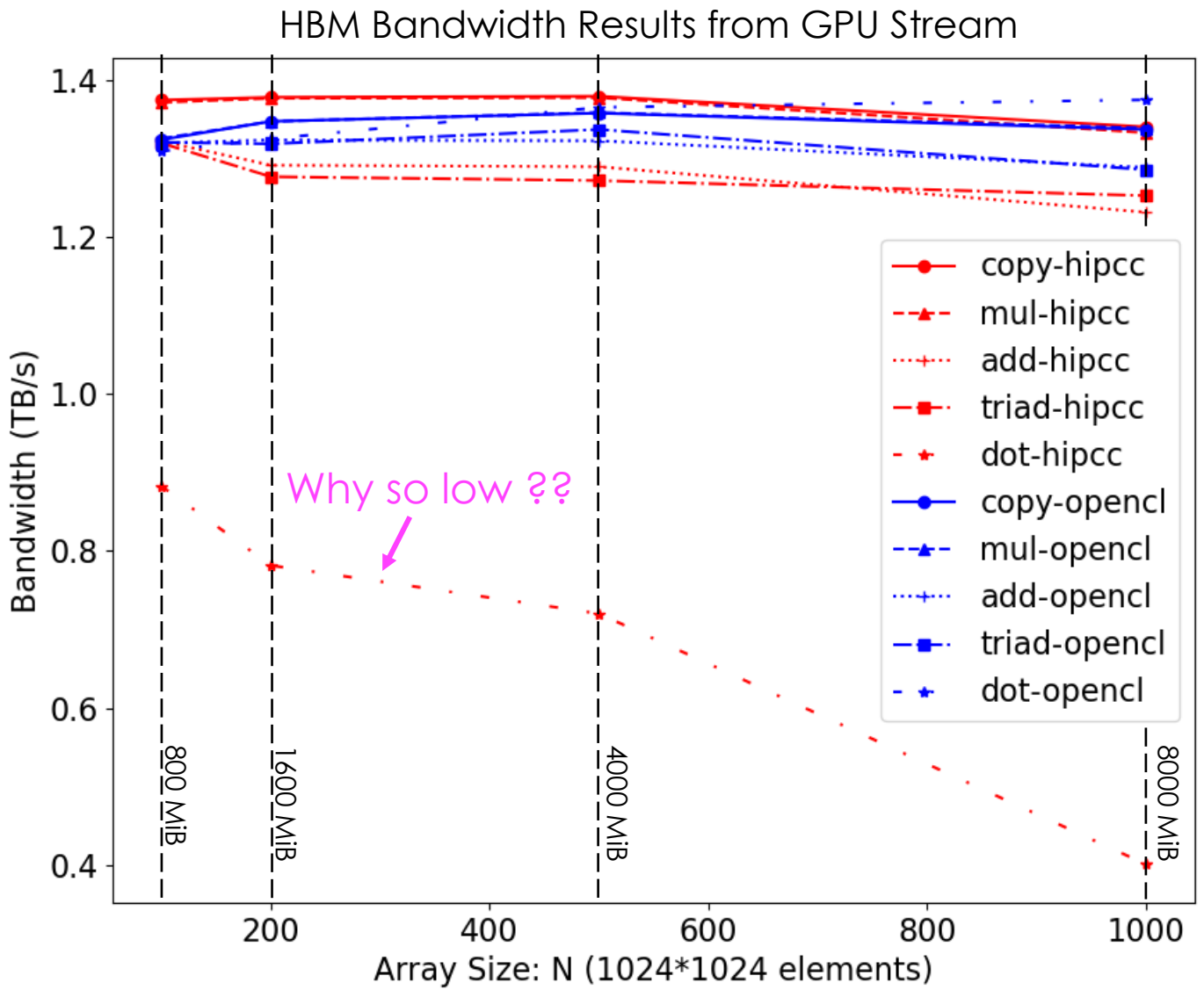
Memory per array is `N*8` bytes
Total memory is `3*N*8` bytes
(where `N` is array size)

GPU HBM Bandwidth

GPU Stream Test

Copy	`A[i] = B[i]`
Scale	`A[i] = n*B[i]`
Add	`A[i] = B[i] + C[i]`
Triad	`A[i] = B[i] + n*C[i]`
Dot	`A[i] = B[i]*C[i]`

- Peak HBM bandwidth: 1.6 TB/s
- Not sure why hipcc dot bandwidth is so low when OpenCL is not…



NOTES:

```
BabelStream
Version: 4.0
Implementation: HIP
Running kernels 100 times
Precision: double
Using HIP device
Driver: 50120531
```

Memory per array is `N*8` bytes
Total memory is `3*N*8` bytes
(where `N` is array size)

GPU HBM Bandwidth

GPU Stream Test

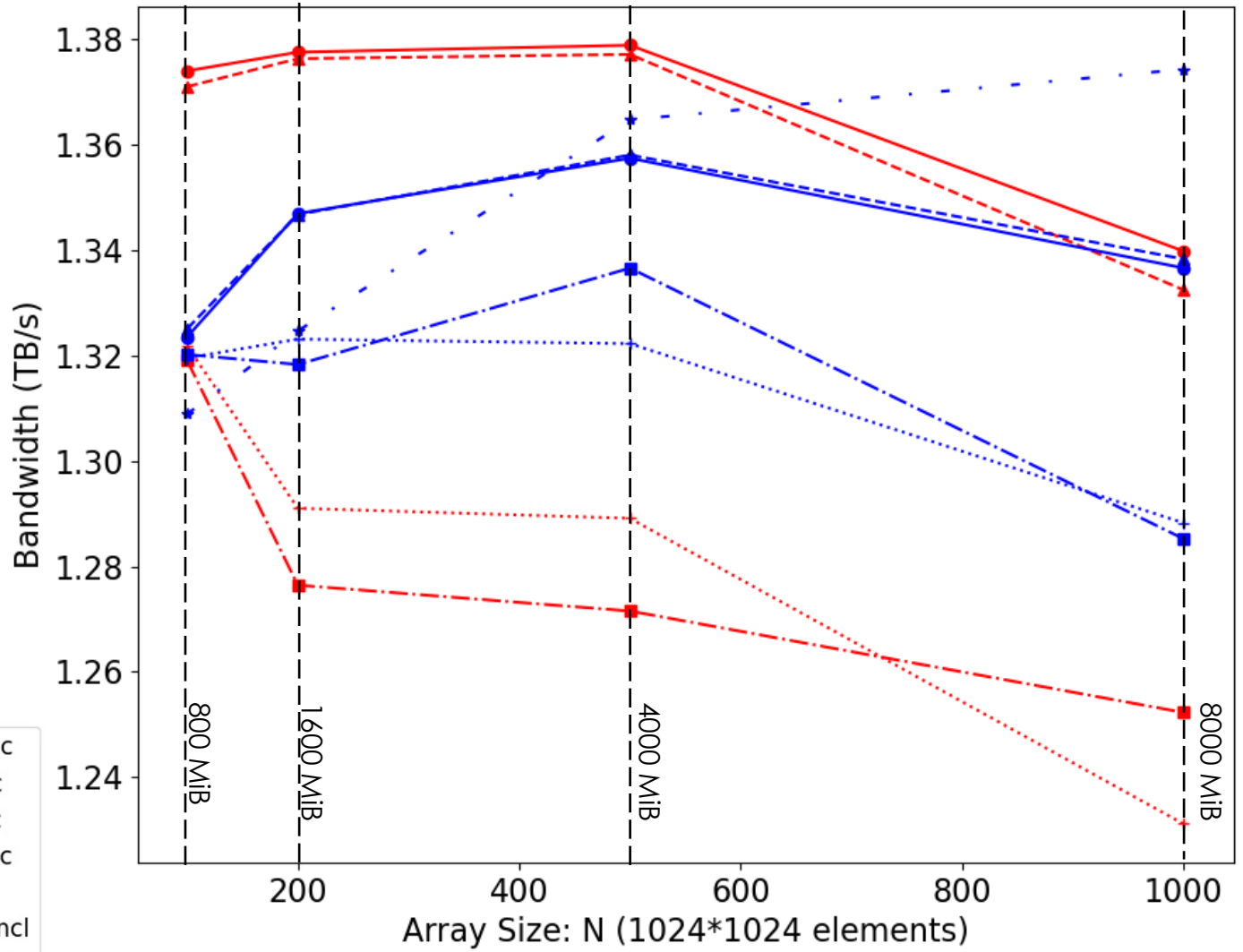
Copy	`A[i] = B[i]`
Scale	`A[i] = n*B[i]`
Add	`A[i] = B[i] + C[i]`
Triad	`A[i] = B[i] + n*C[i]`
Dot	`A[i] = B[i]*C[i]`

- Peak HBM bandwidth: 1.6 TB/s
- Not sure why hipcc dot bandwidth is so low when OpenCL is not…
- Observed DRAM bandwidth: 77-86% of peak

NOTES:

```
BabelStream
Version: 4.0
Implementation: HIP
Running kernels 100 times
Precision: double
Using HIP device
Driver: 50120531
```

HBM Bandwidth Results from GPU Stream



Memory per array is `N*8` bytes
Total memory is `3*N*8` bytes

(where `N` is array size)

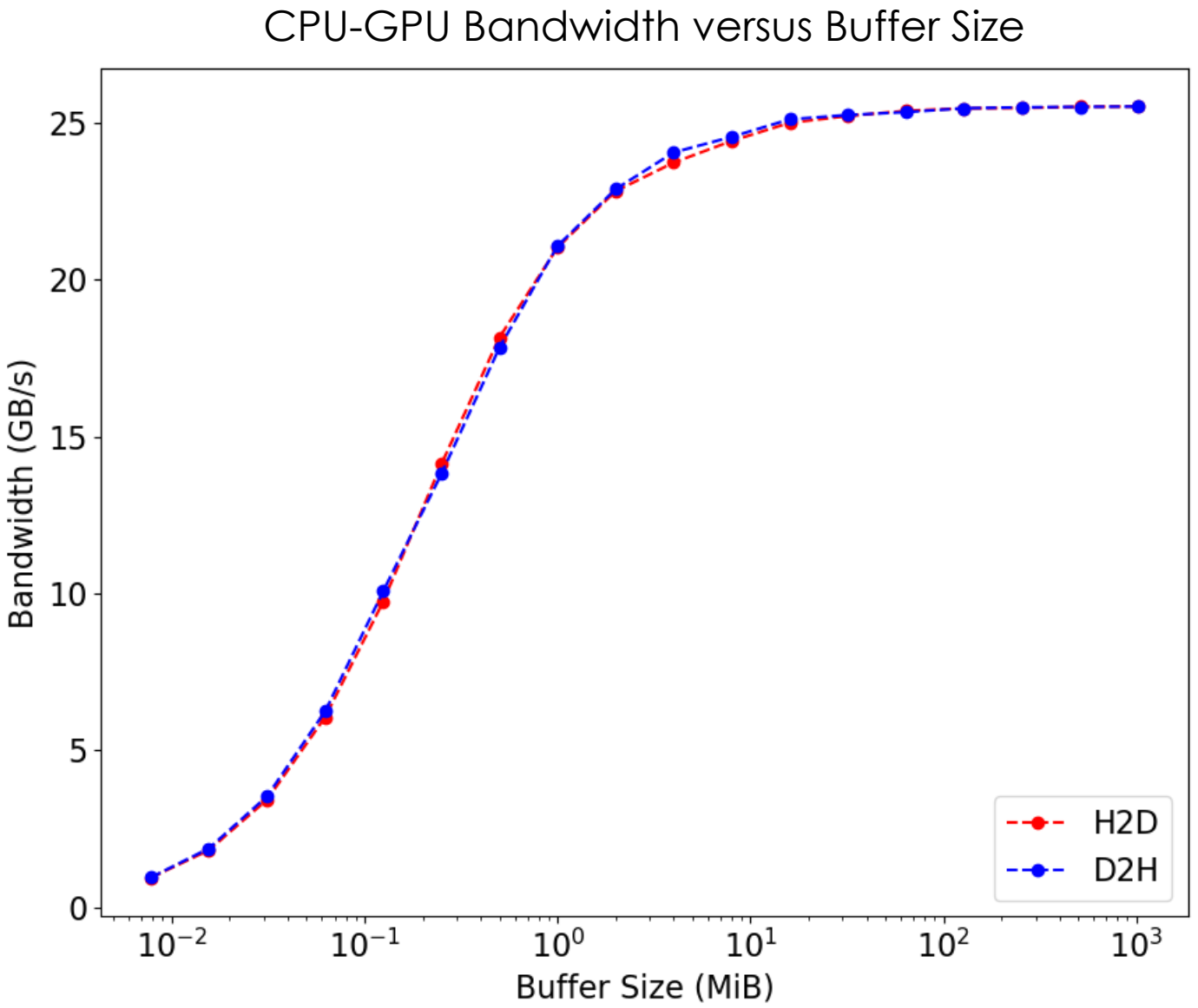
CPU-GPU Bandwidth

https://github.com/tom-papatheodore/h2d_d2h_bandwidth

- Simple bandwidth test
- Measures host-to-device and device-to-host bandwidth using 50 iterations of `hipMemcpy` in each direction.
- Timing measured with `hipEvent`

Peak CPU-GPU bandwidth: 36 GB/s

Max. observed bandwidth: 25.6 GB/s (71% of peak)



GPU-GPU Bandwidth

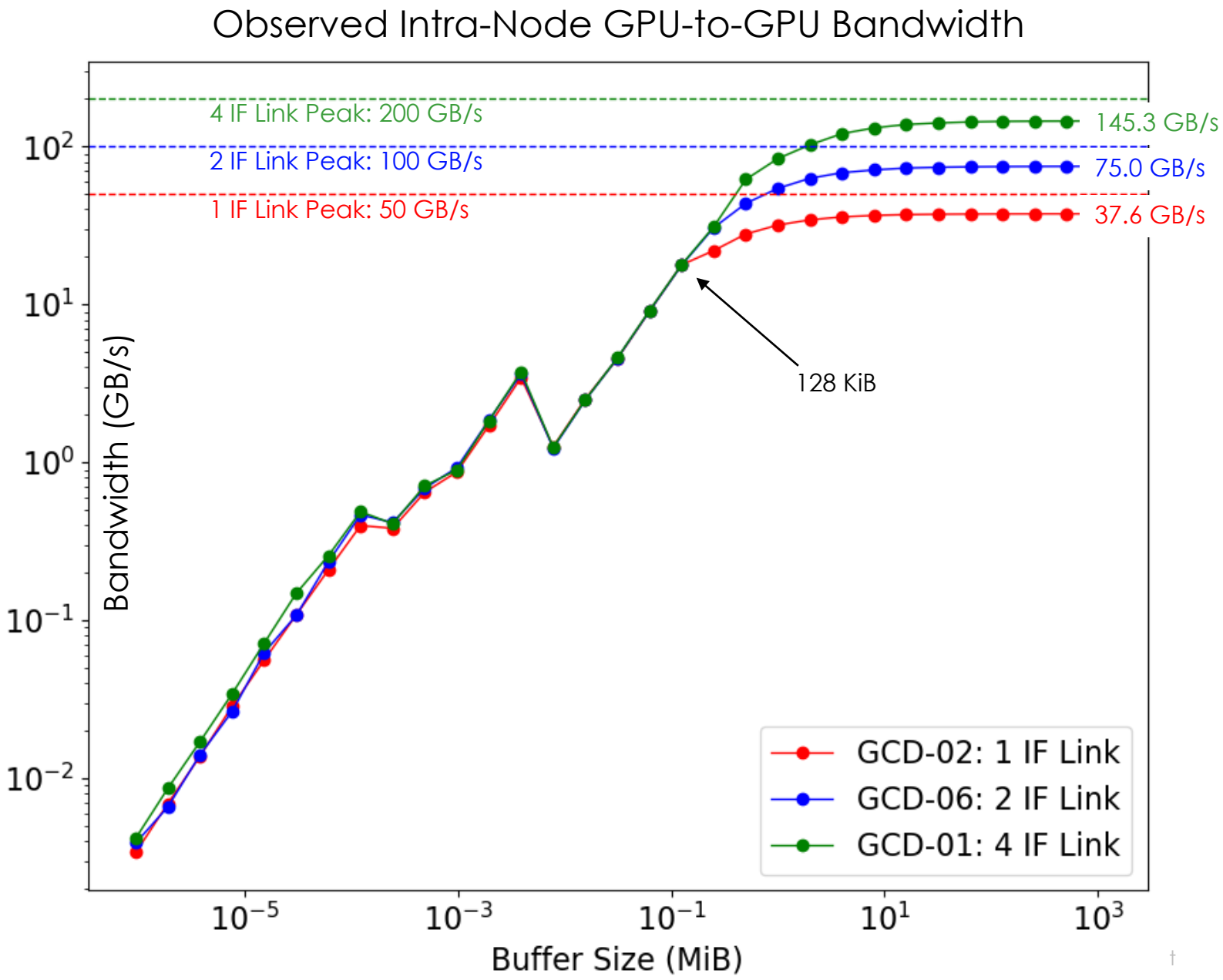
OSU Microbenchmarks: **`osu_bw`**

https://mvapich.cse.ohio-state.edu/benchmarks/

Measure BW between each GPU pair on a node.

Recall from the Frontier node diagram, there can be 1, 2, or 4 InfinityFabric (IF) links between GPUs on a node.

IF Links	BW (GB/s)
1	37.6 (75.2% of peak)
2	75.0 (75.0% of peak)
4	145.3 (72.7% of peak)



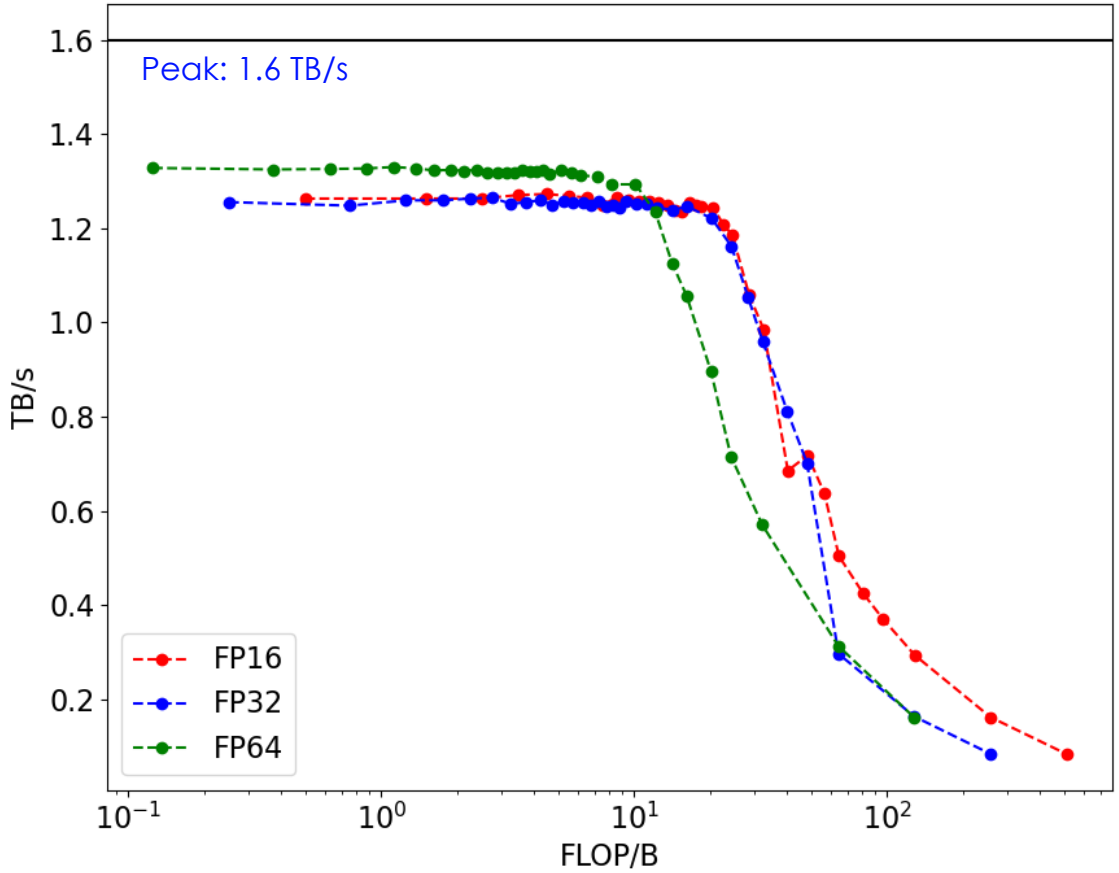
Mixbench

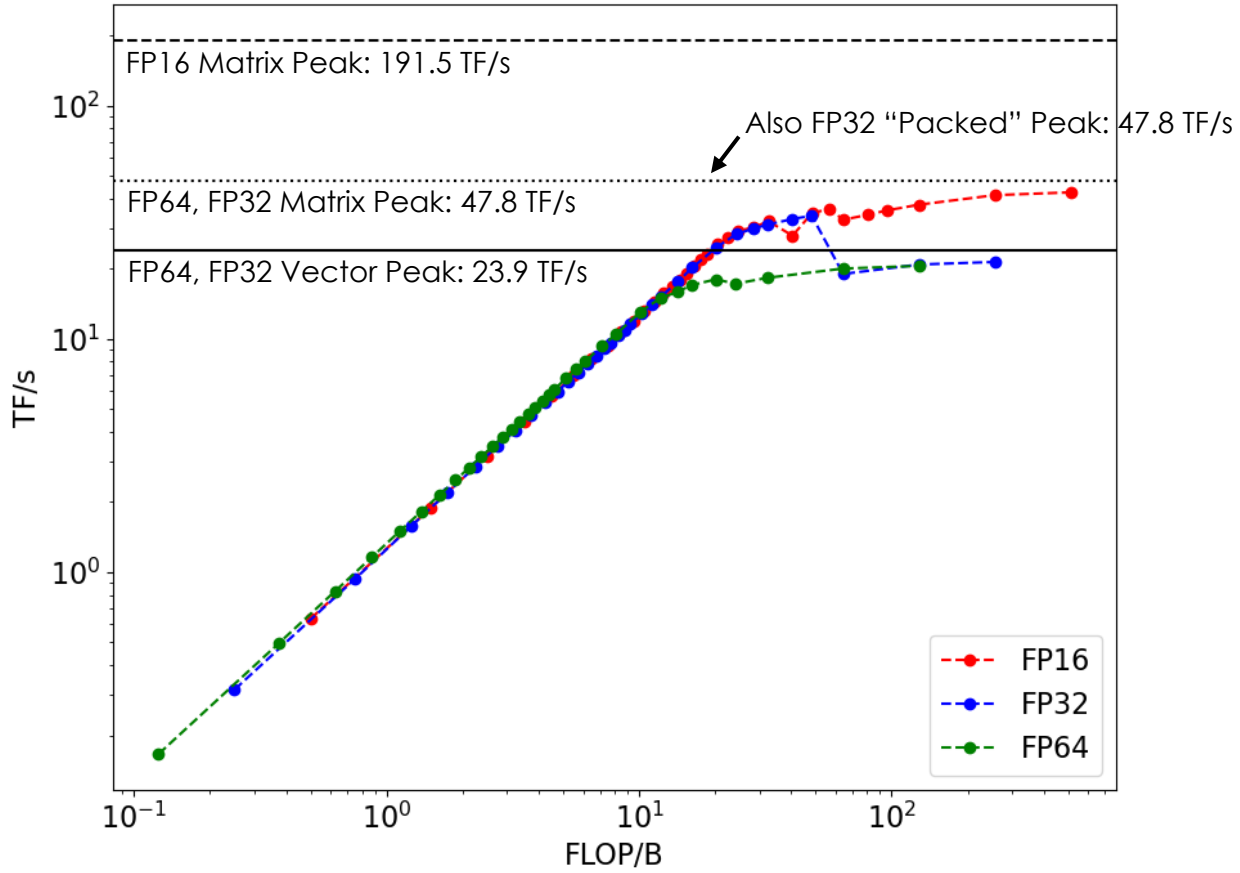
https://github.com/ekondis/mixbench

Measures FLOP/s and GB/s vs FLOP/B

- Buffer size stays fixed while compute iterations increase
- FP64, FP32, and FP16 results

```
---------------------- Device specifications ----------------------
Device:
CUDA driver version: 50322.61
GPU clock rate:      1700 MHz
WarpSize:            64
L2 cache size:       8192 KB
Total global mem:    65520 MB
Total SPs:           7040 (110 MPs x 64 SPs/MP)
Compute throughput:  23936.00 GFlops (theoretical single precision FMAs)
Memory bandwidth:    1638.40 GB/sec
-------------------------------------------------------------------
Total GPU memory 68702699520, free 68702699520
Buffer size:          256MB
Trade-off type:       compute with global memory (block strided)
Elements per thread:  8
Thread fusion degree: 1
```





For details on the peak values and packed FP32 operations: https://www.amd.com/system/files/documents/amd-cdna2-white-paper.pdf

GPU Compute – Dense DGEMM

gpu_xgemm

https://github.com/tom-papatheodore/gpu_xgemm

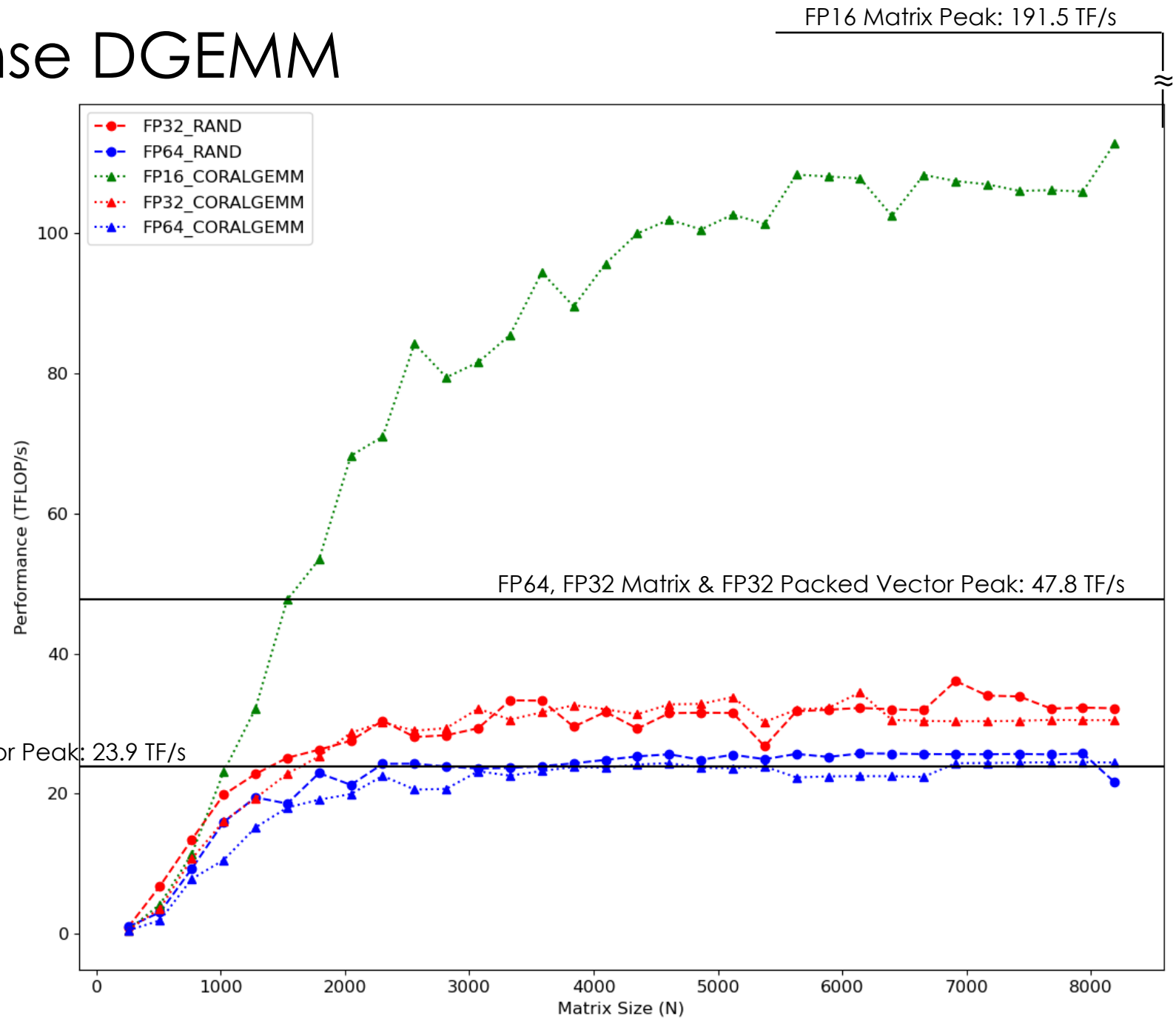
- FP32/64_RAND

CORALGEMM

https://github.com/AMD-HPC/CoralGemm

- FP16/32/64_CORALGEMM

Both of these tests use hipBLAS.



Values for peak obtained from Table 1 of MI250X white paper: https://www.amd.com/system/files/documents/amd-cdna2-white-paper.pdf

Expected Performance (Summit-to-Frontier)

Performance expectations on Frontier relative to Summit

- Is your application GPU-bound or data-transfer-bound?
  - If GPU-bound, is it compute-bound or memory-bound?
    - Compute-bound: compute performance
    - Memory-bound: HBM, HBM bandwidth, L1 cache
    - This is where kernel profiling can be helpful
  - If data-transfer-bound, consider CPU-to-GPU bandwidth in table.

	V100	MI250X (GCD)	MI250X (GCD) / V100	Summit Node	Crusher Node	Crusher Node / Summit Node
Compute performance	~7.8 TF	~23.9 TF	~3.1X	~47 TF	~191 TF	~4.1X
HBM	16 GB	64 GB	4X	96 GB	512 GB	~5.3X
HBM bandwidth	0.9 TB/s	1.6 TB/s	~1.8X	5.4 TB/s	12.8 TB/s	~2.4X
CPU-to-GPU bandwidth	50 GB/s	36 GB/s	~0.7X	300 GB/s	288 GB/s	~0.96X
L1 cache	Up to 128 KB	16 KB	0.125X – 0.5X			
L2 cache	6 MB	8 MB	1.3X			
Network bandwidth				25 GB/s	100 GB/s	4X

Summary

- Regarding raw FP64 FLOP/s, it's *possible* to achieve most of the peak performance out of the MI250X – although this is of course dependent on your application.
  - And in some cases, perhaps even more than expected when libraries take advantage of the matrix cores.
- Observed DRAM bandwidth: 82-90% of peak
- Observed DRAM bandwidth: 77-86% of peak
- Observed CPU-GPU bandwidth: 71% of peak
- Observed GPU-GPU bandwidth: 72-75% of peak
- When comparing Frontier results to Summit (or another system), it's important to understand how all the differences in speeds-and-feeds affect **your application**.
- Other considerations:
  - Warp size is 64 on MI250X, but 32 on V100.
  - L1 cache is smaller on MI250X
  - Register usage can affect performance
  - HW atomics must be enabled to use: https://docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html#floating-point-fp-atomic-operations-and-coarse-fine-grained-memory-allocations
  - GPU page migration w/managed memory must be enabled to use: https://docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html#enabling-gpu-page-migration

Questions?

papatheodore@ornl.gov

OAK RIDGE National Laboratory