Programming Techniques for Supercomputers

Friedrich-Alexander-Universität
Erlangen-Nürnberg

GPU Programming – Performance

Sebastian Kuckuk

Erlangen National High Performance Computing Center (NHR@FAU)

NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

GPU Programming

GPU Benchmarks

Performance Modelling – General Approach

1. Obtain relevant metrics for relevant code regions

   - Execution time
   - Bytes read/ written
   - FLOPS performed
   - And many more …

   Somewhere between kernels/ loops and whole application; usually limited to ‘meaningful’ work

2. Relate observed performance to theoretical limits
   - … or to measured limits – see next slides

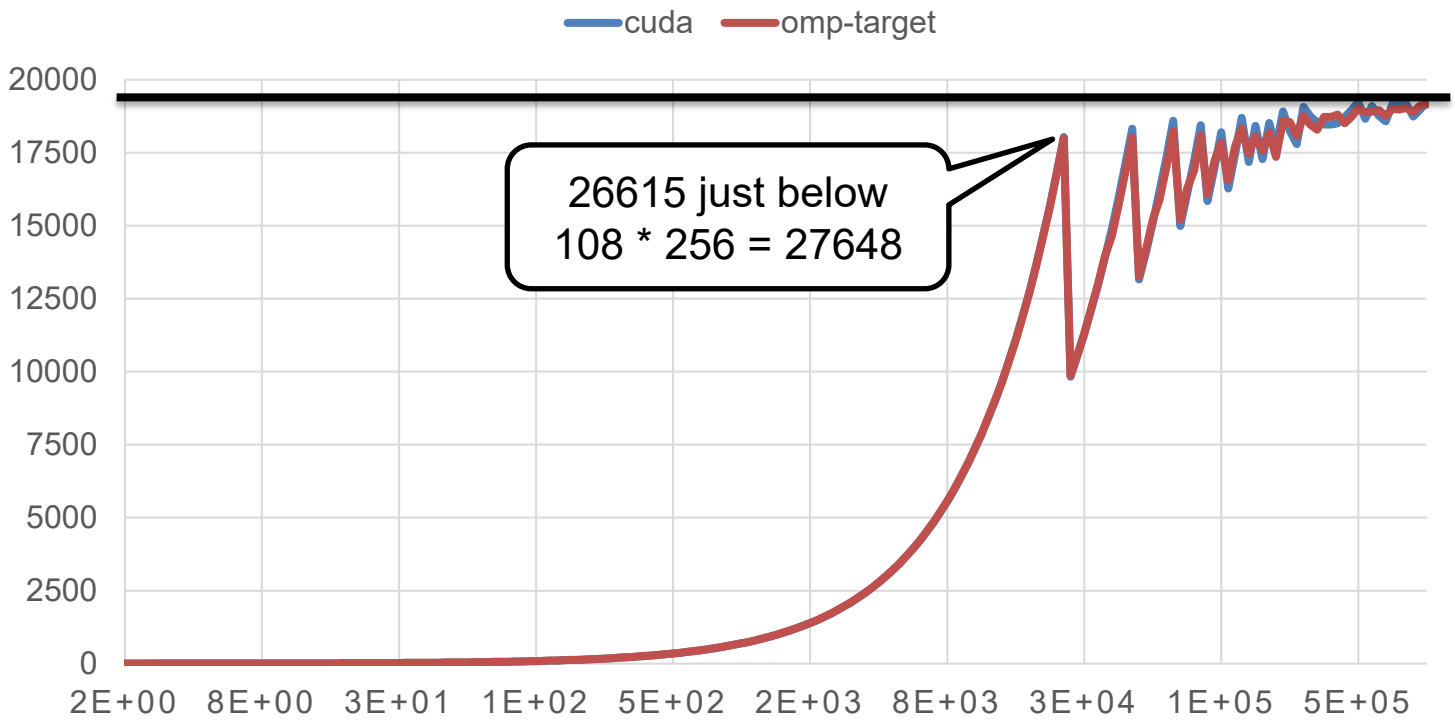
3. Figure out where optimization is possible/ reasonable
   - Try to hit at least one bottleneck, then shift to another bottleneck

Micro Benchmarks

- Motivation: theoretical limits can be too optimistic, micro benchmarks give a more realistic expectation
- Two examples: computational performance and sustained bandwidth
- This also represents a manual approach to performance modelling – doing explicit timing combined with manual code analysis
- The following results are obtained on a single A100 80GB GPU in Alex

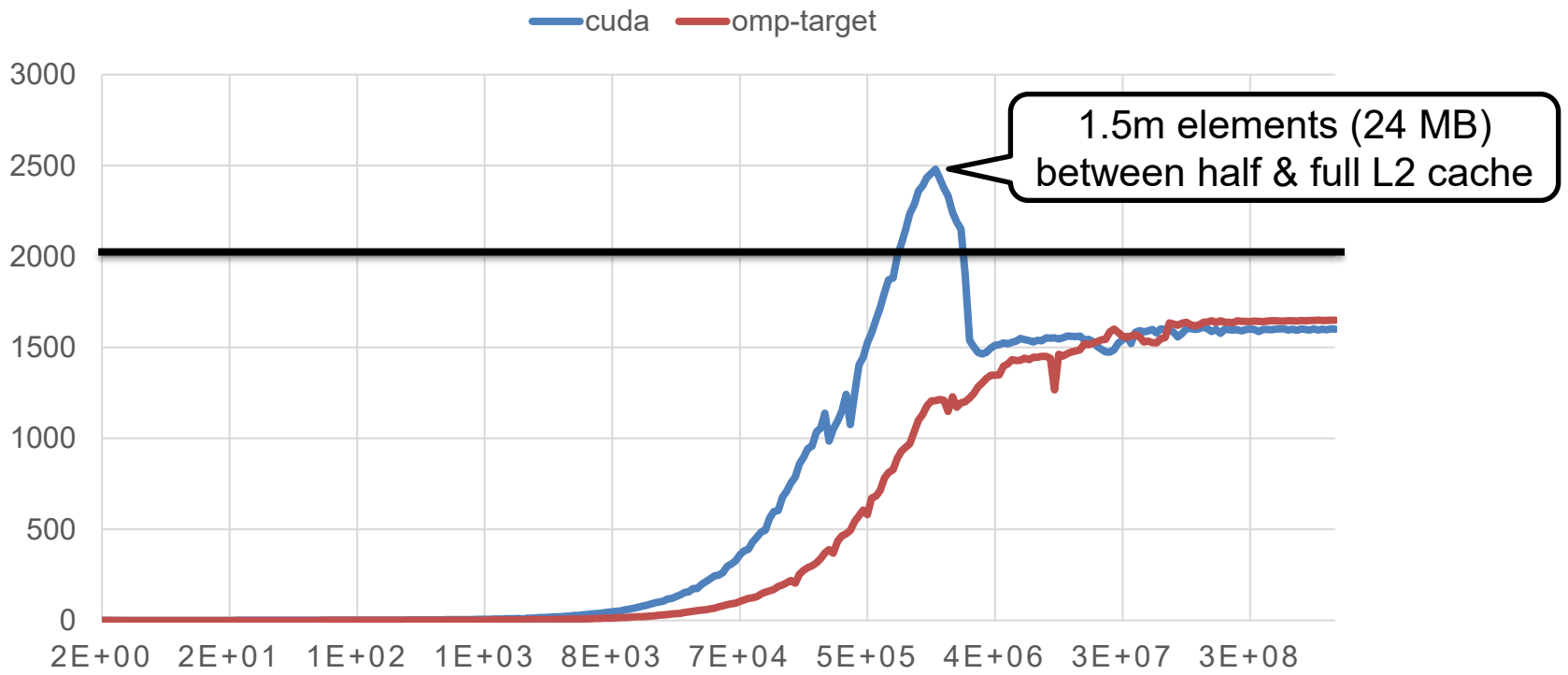
FMA Benchmark

A100, FMA, GFLOP/s over number of computational elements



Stream Benchmark

A100, stream, bandwidth in GB/s over number of array elements

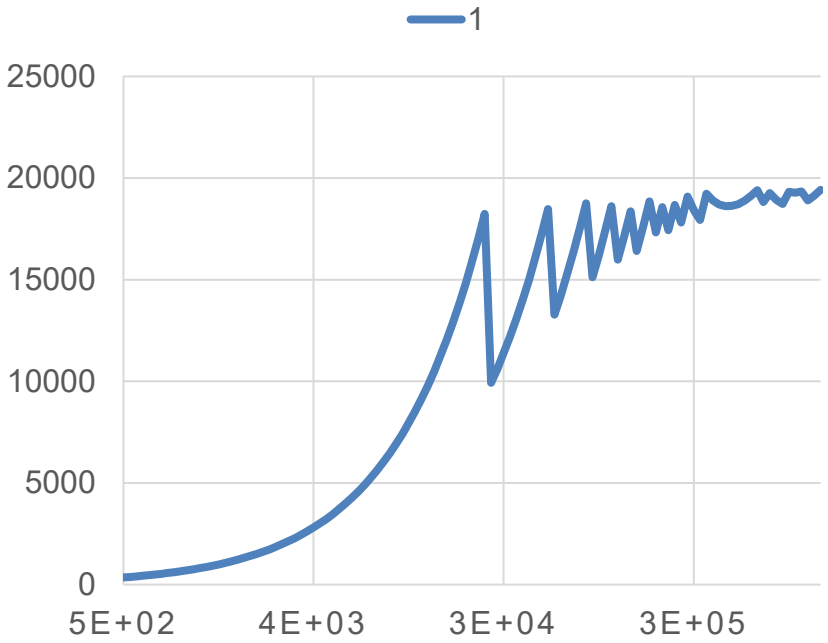


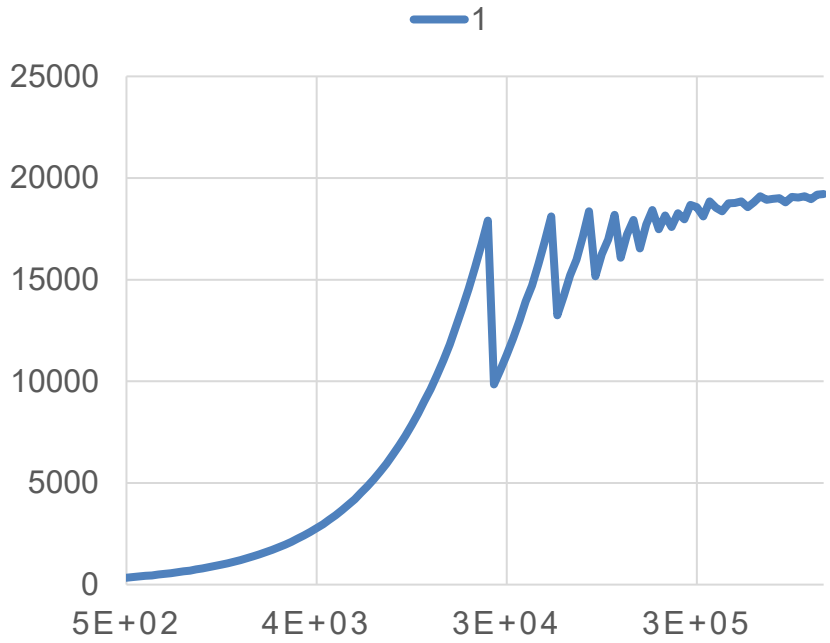
Benchmark Evaluation

- FMA performance is close to theoretical peak
- Stream performance is not perfect
  - Potentially different results for different data access patterns (#load and #store operations per thread)
  - Potentially different results for different data types
- What about non-uniform operations?
  - Strided Stream – each thread accesses data with an added stride
  - Strided FMA – each thread that is divisible by the stride computes

Strided FMA Benchmark

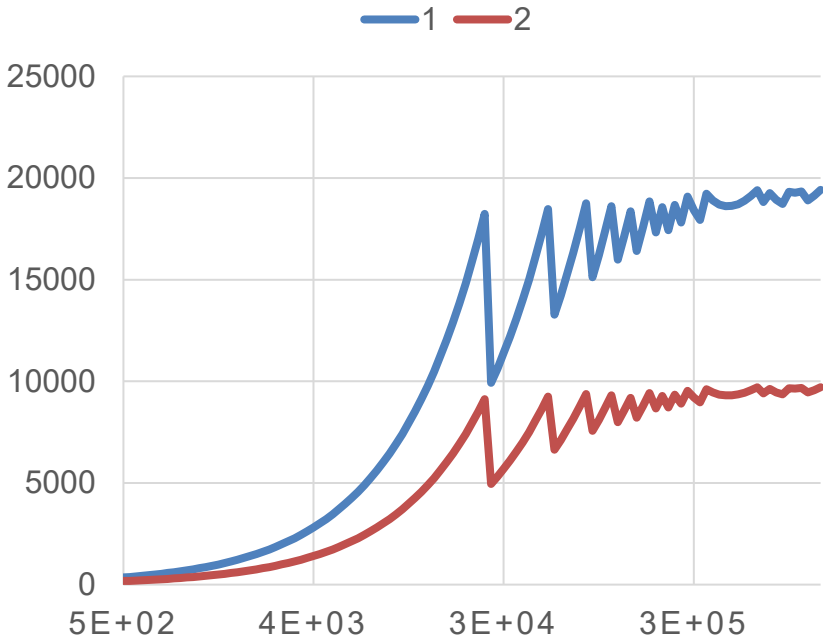
A100, strided FMA, GFLOP/s over number of computational elements
CUDA (left) and OpenMP target (right)

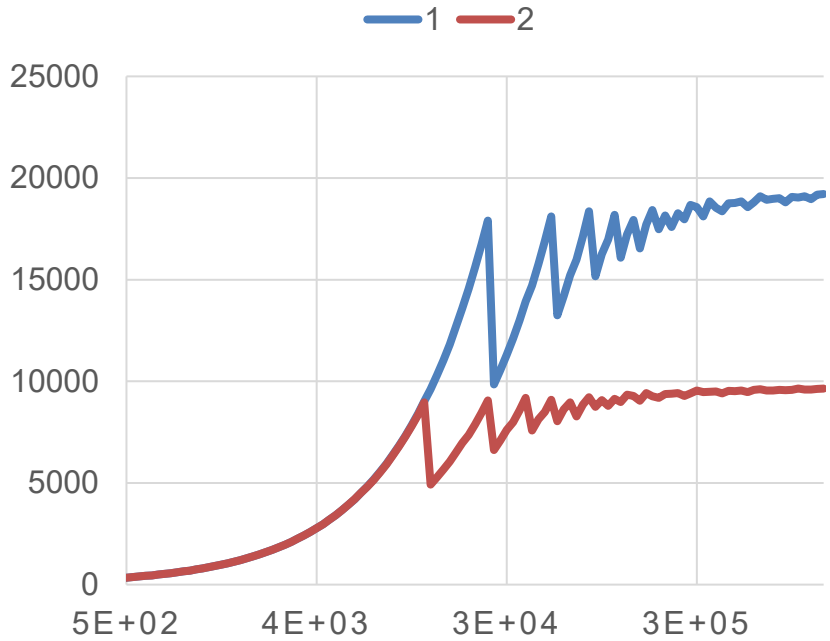




Strided FMA Benchmark

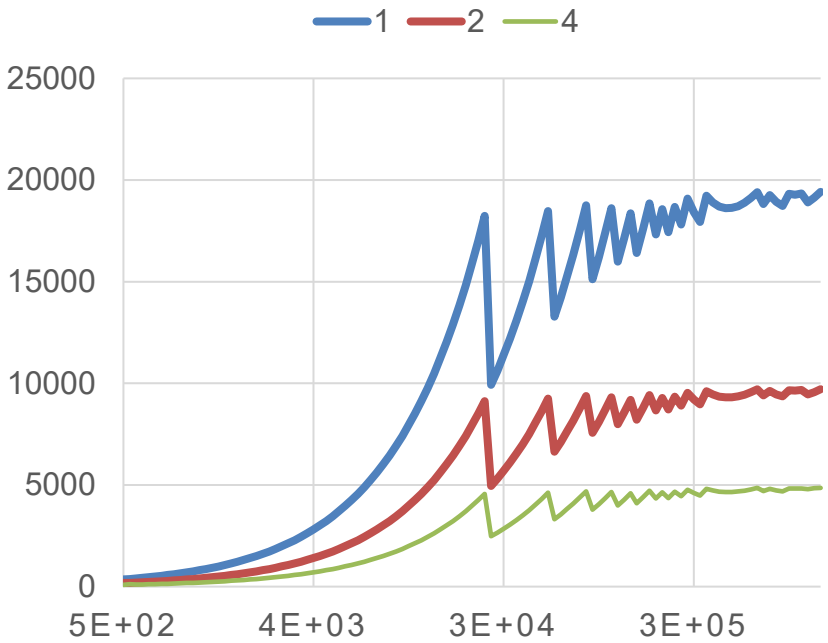
A100, strided FMA, GFLOP/s over number of computational elements
CUDA (left) and OpenMP target (right)

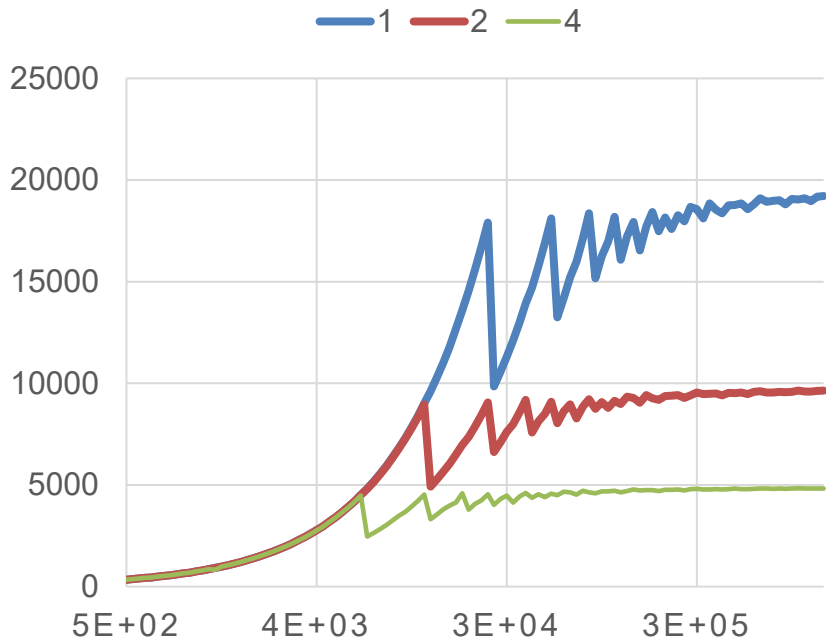




Strided FMA Benchmark

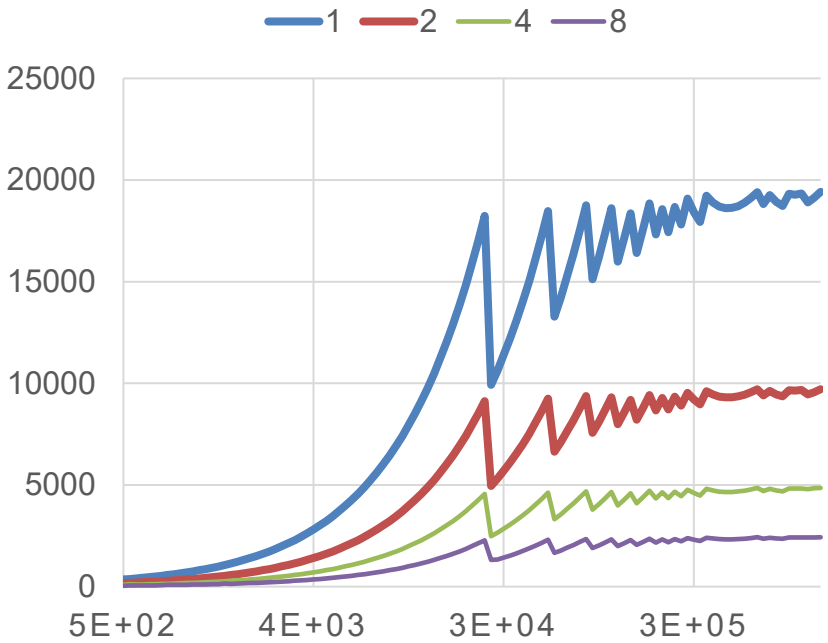
A100, strided FMA, GFLOP/s over number of computational elements
CUDA (left) and OpenMP target (right)

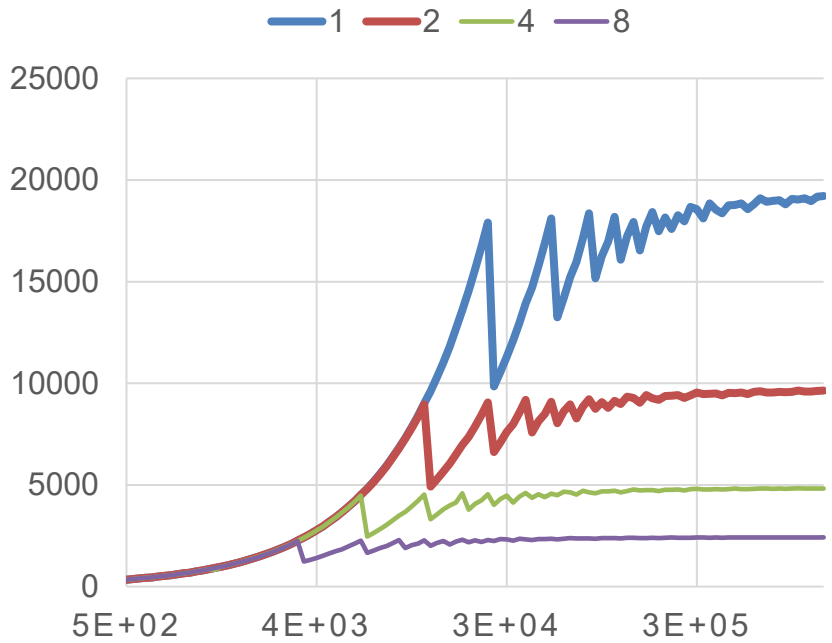




Strided FMA Benchmark

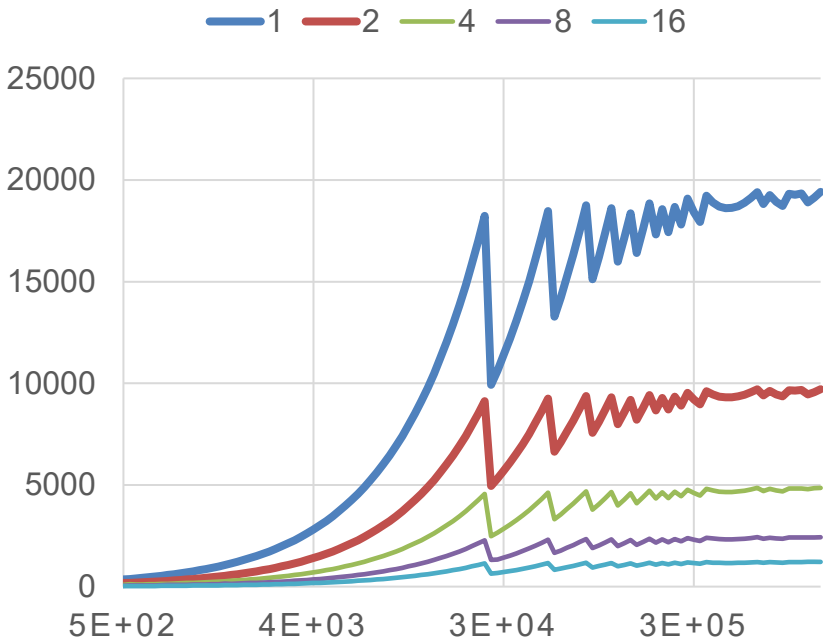
A100, strided FMA, GFLOP/s over number of computational elements
CUDA (left) and OpenMP target (right)

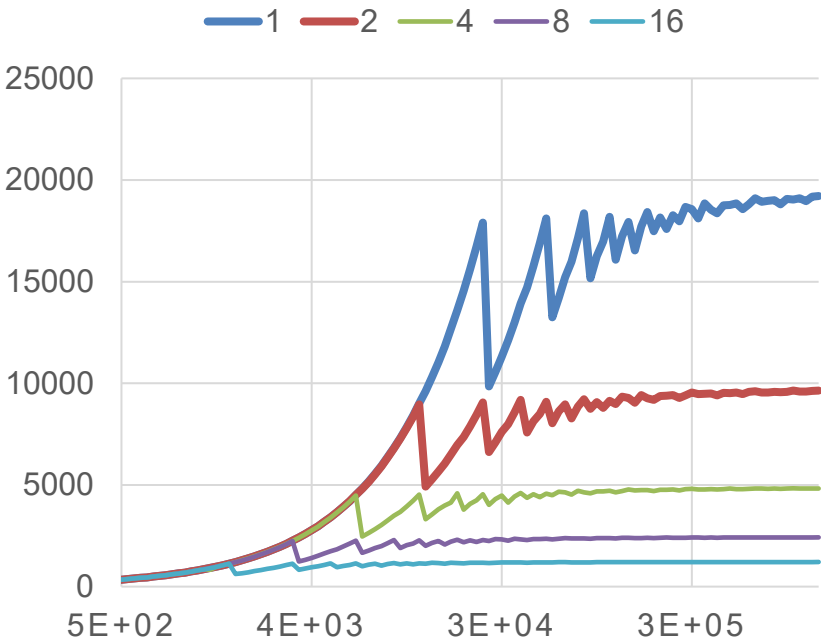




Strided FMA Benchmark

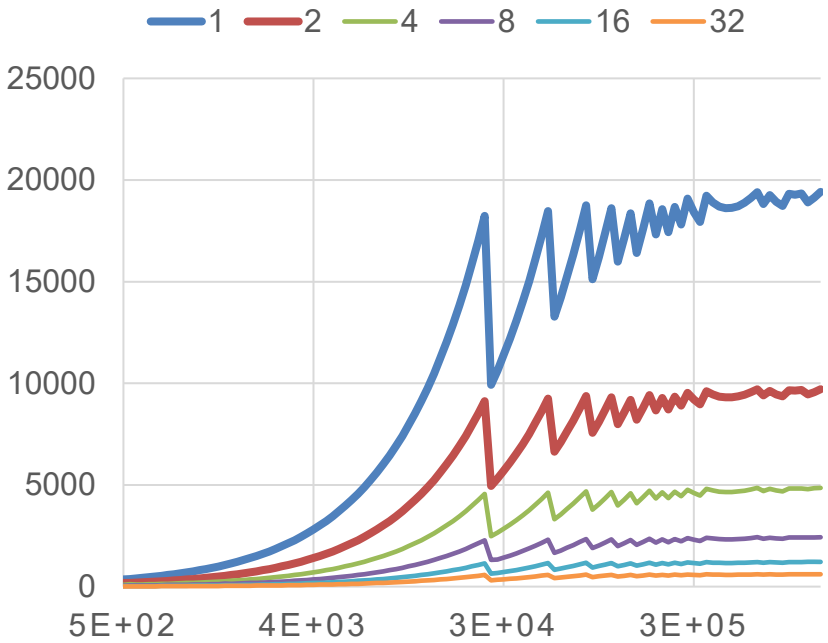
A100, strided FMA, GFLOP/s over number of computational elements
CUDA (left) and OpenMP target (right)

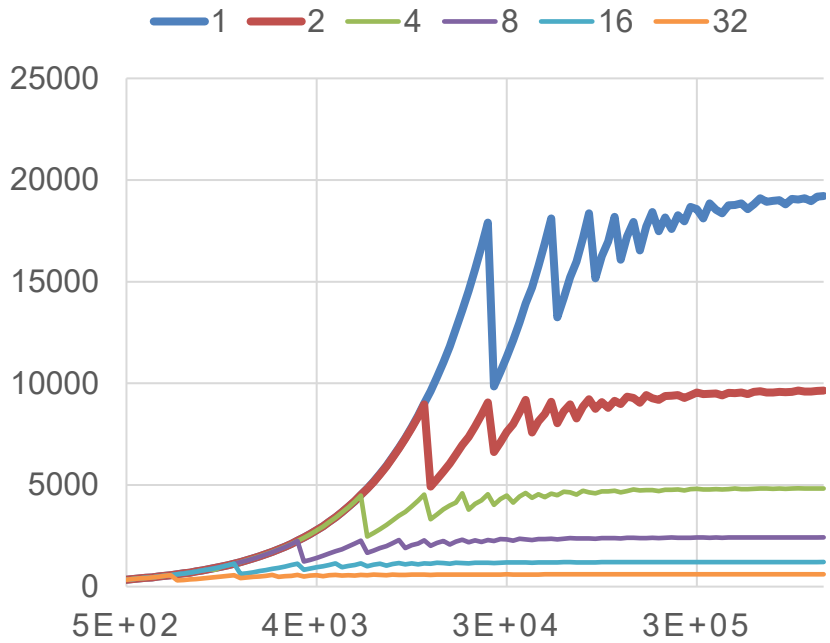




Strided FMA Benchmark

A100, strided FMA, GFLOP/s over number of computational elements
CUDA (left) and OpenMP target (right)





Memory Coalescing

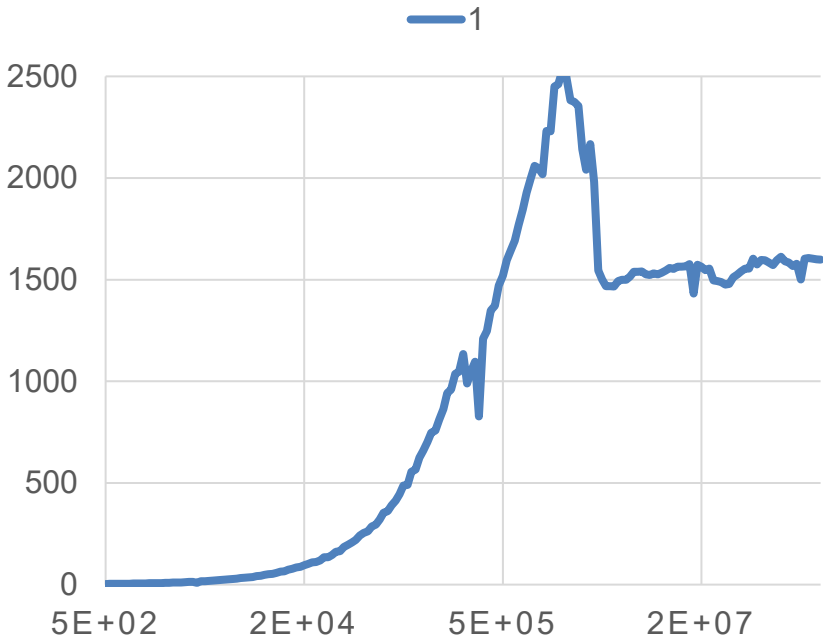
- Simple copy benchmark with strided access pattern

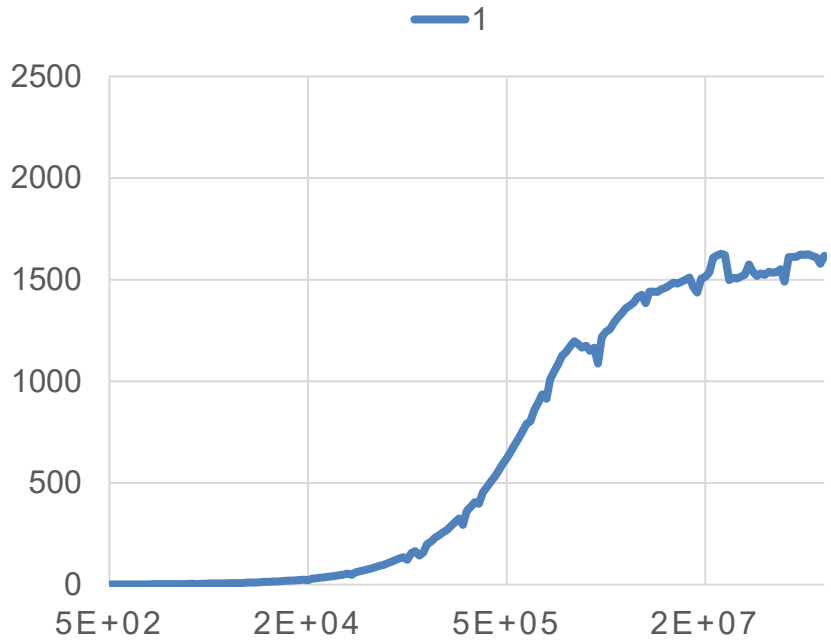
```
__global__ void copy(double *src, double *dest, size_t nx, int stride) {
  size_t i = blockIdx.x * blockDim.x + threadIdx.x;

  if (i < nx)
    dest[stride * i] = src[stride * i];
}
```

- All results are obtained on a single A100 80GB GPU in Alex

Strided Stream Benchmark

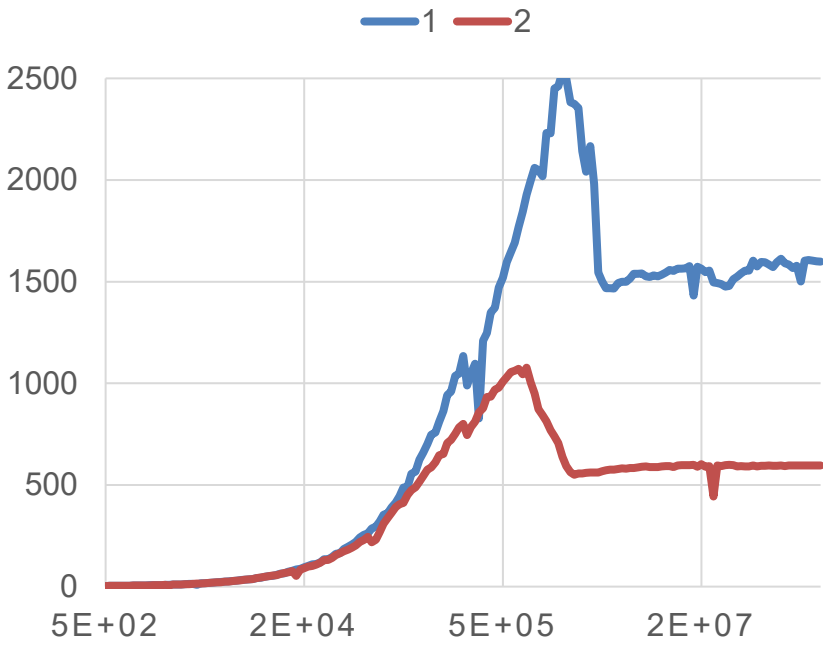
A100, strided stream, bandwidth in GB/s over number of array elements
CUDA (left) and OpenMP target (right)

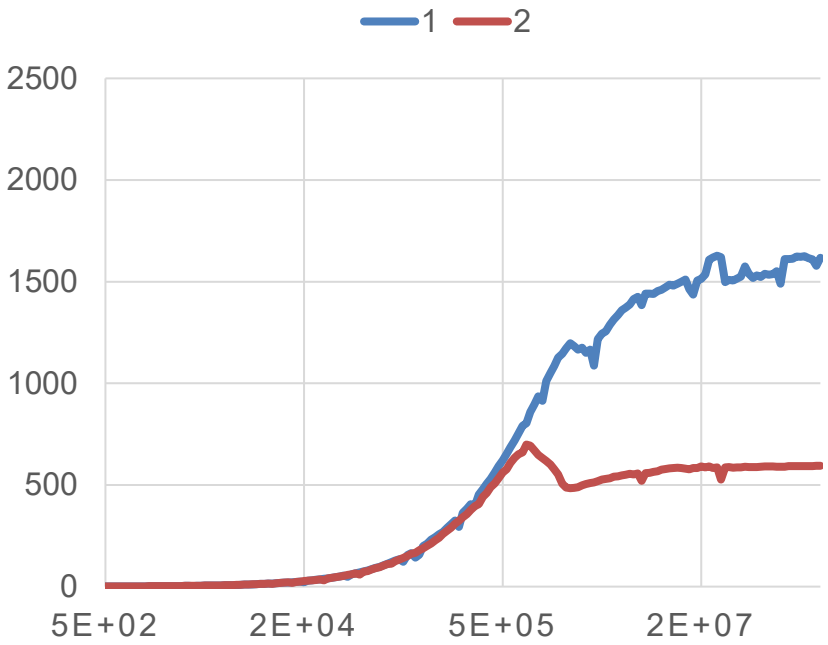




Strided Stream Benchmark

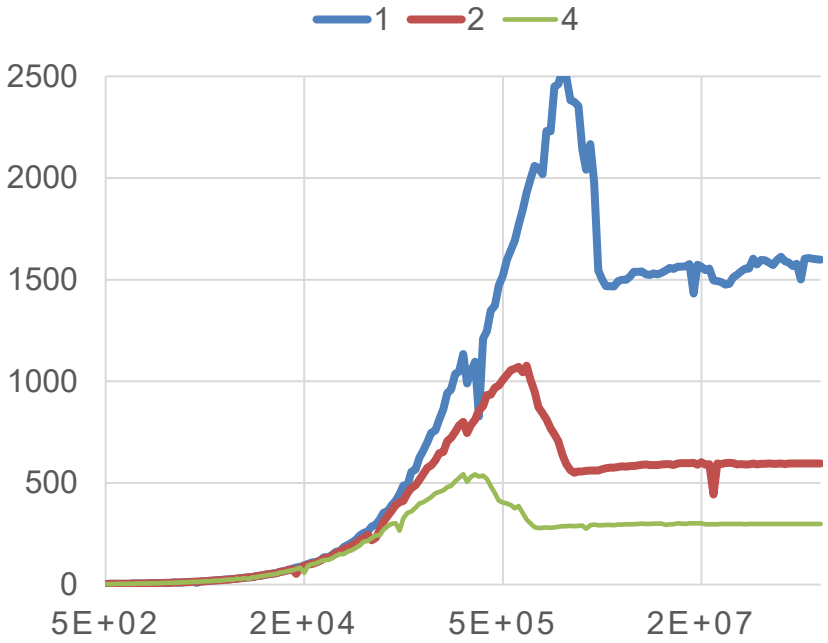
A100, strided stream, bandwidth in GB/s over number of array elements
CUDA (left) and OpenMP target (right)

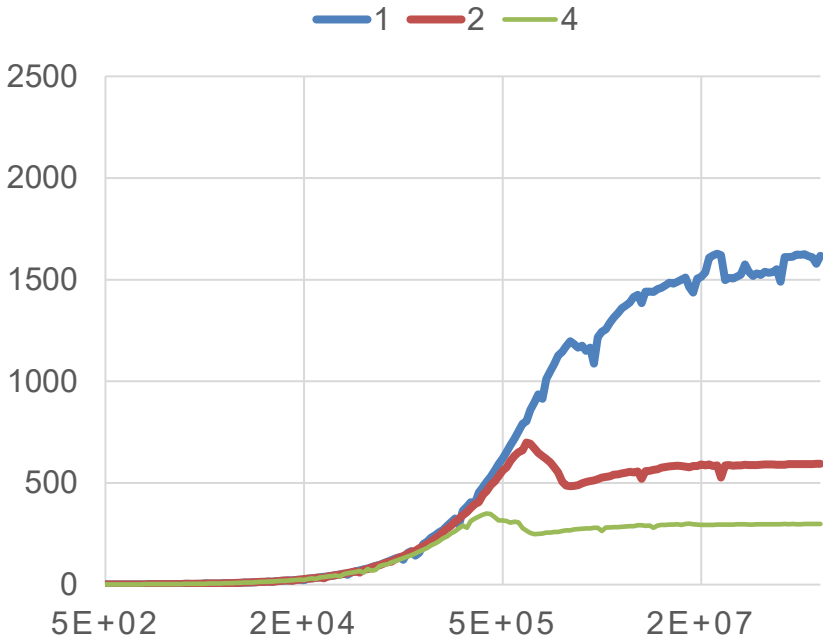




Strided Stream Benchmark

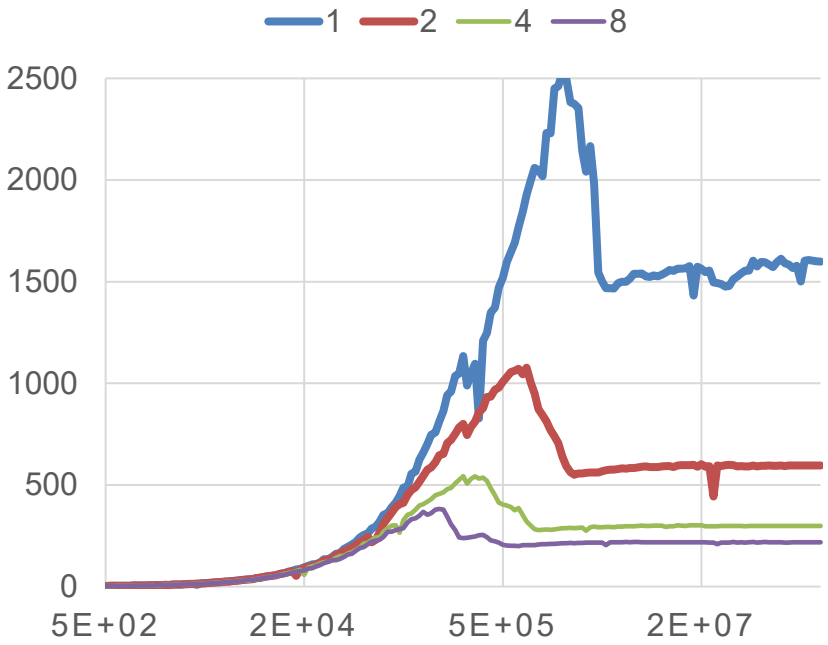
A100, strided stream, bandwidth in GB/s over number of array elements
CUDA (left) and OpenMP target (right)

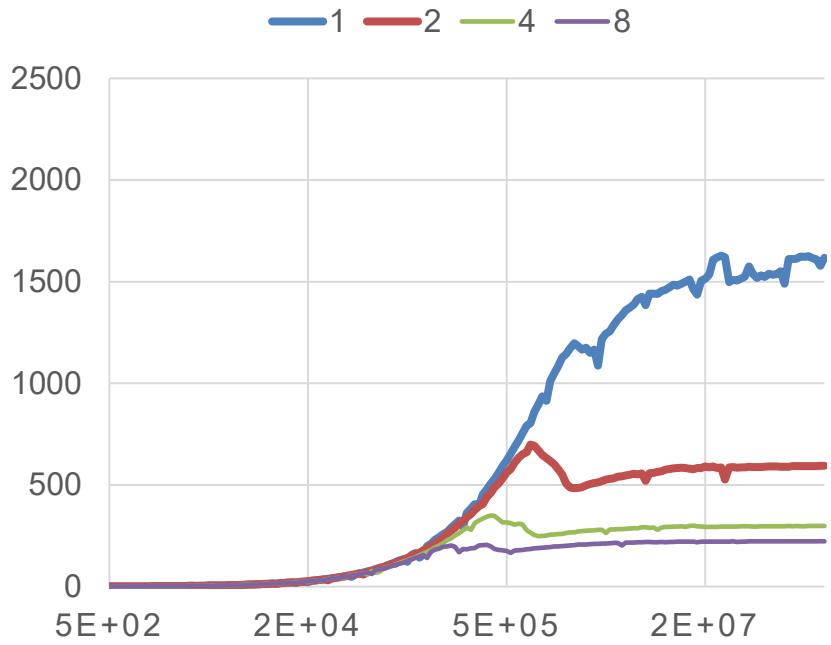




Strided Stream Benchmark

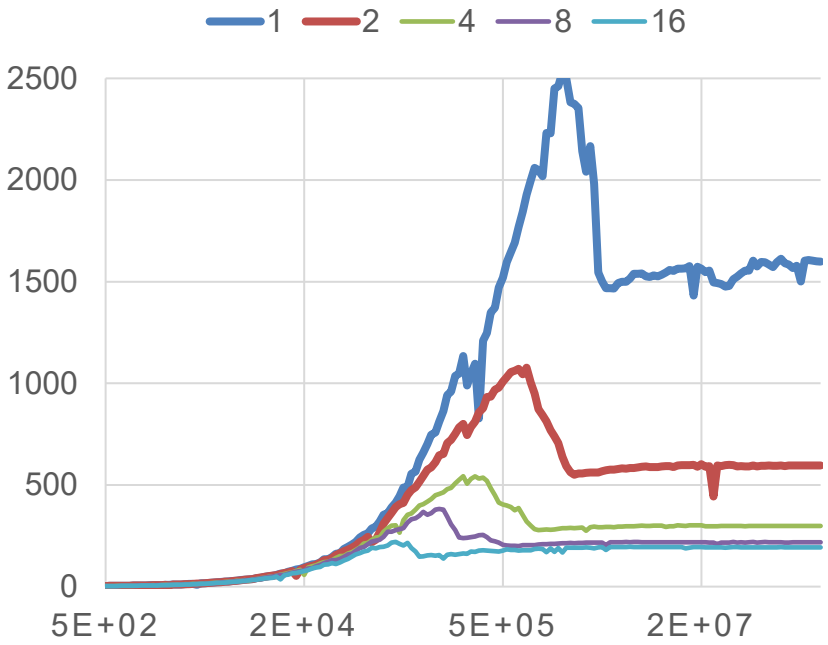
A100, strided stream, bandwidth in GB/s over number of array elements
CUDA (left) and OpenMP target (right)

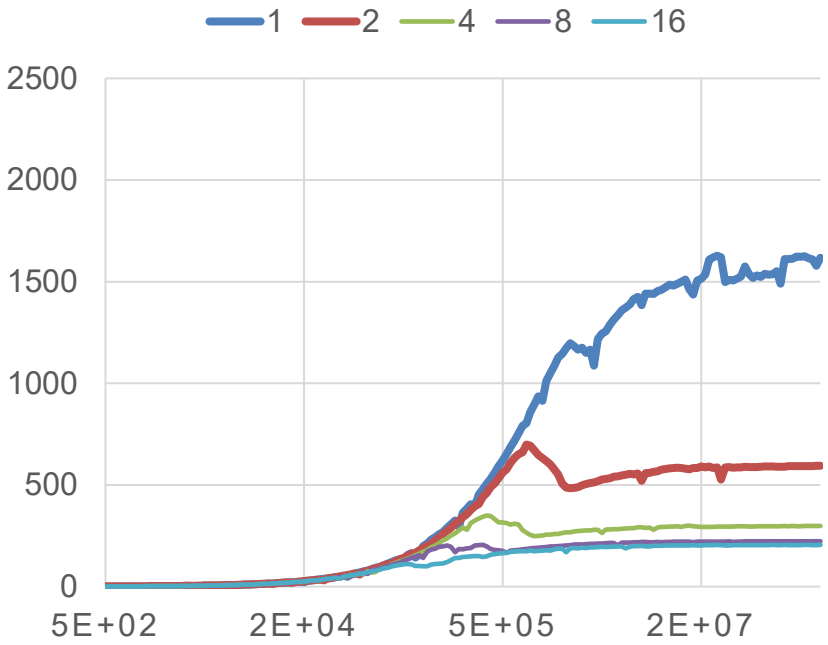




Strided Stream Benchmark

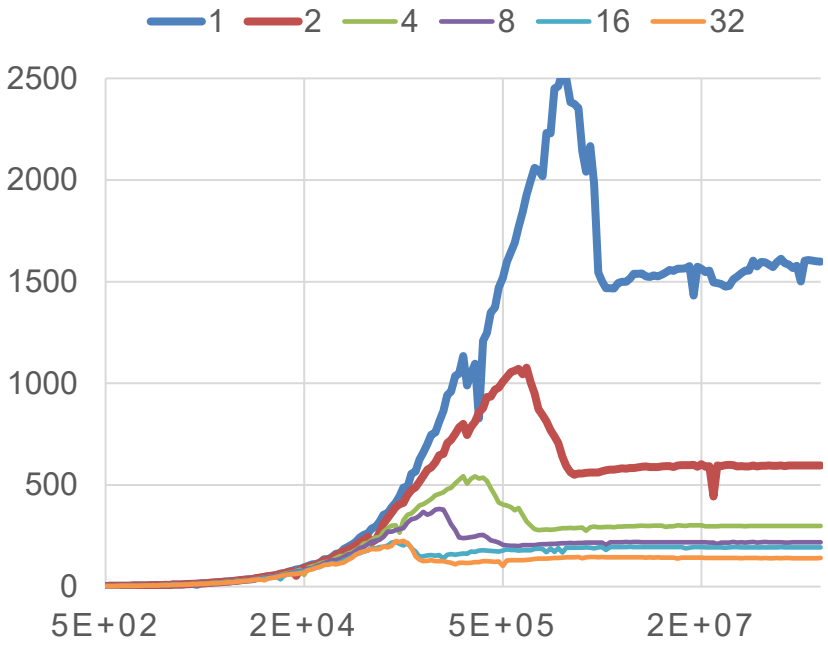
A100, strided stream, bandwidth in GB/s over number of array elements
CUDA (left) and OpenMP target (right)

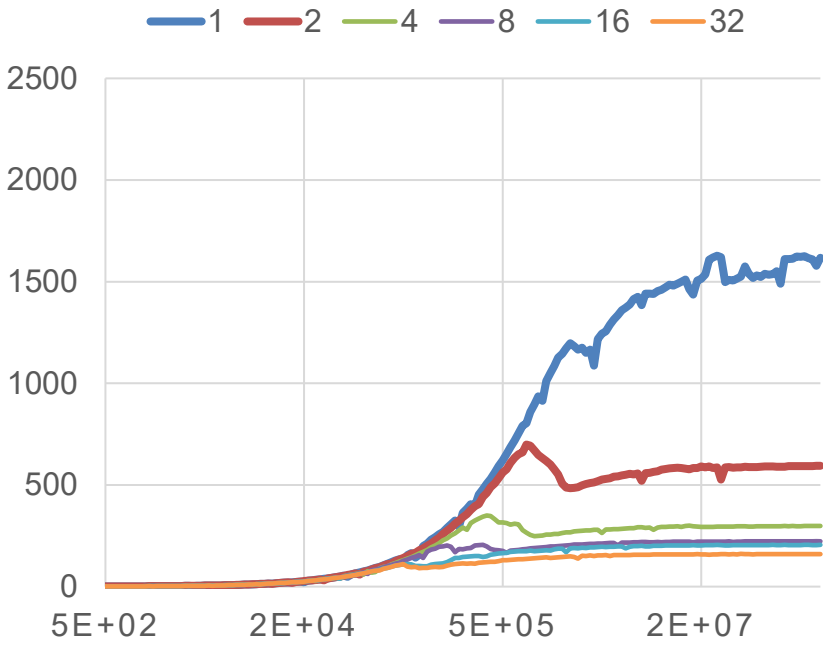




Strided Stream Benchmark

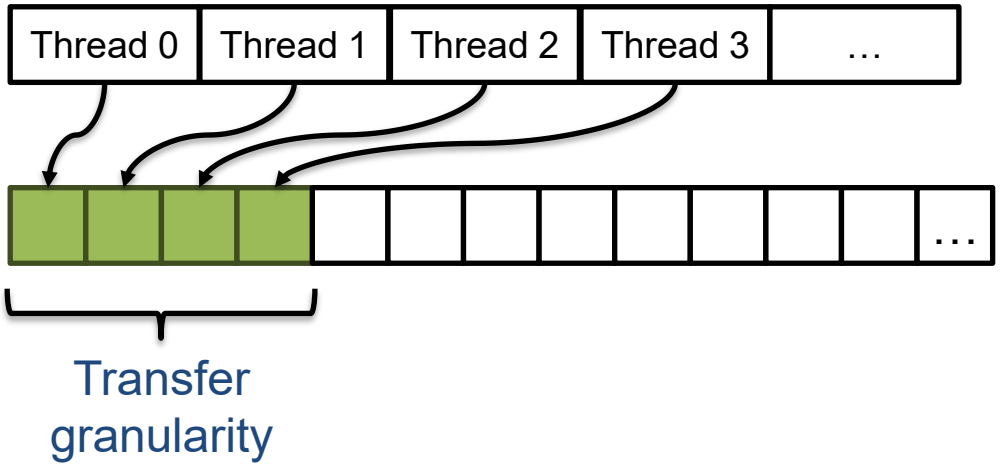
A100, strided stream, bandwidth in GB/s over number of array elements
CUDA (left) and OpenMP target (right)





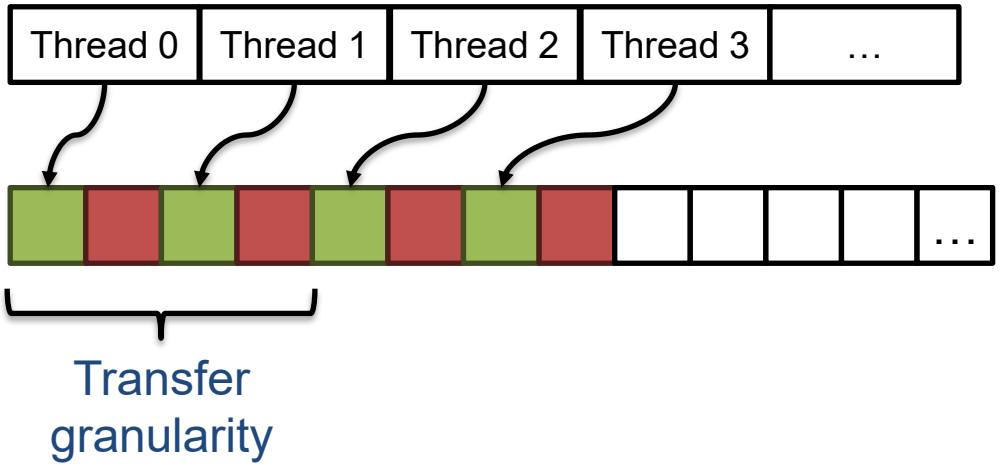
Memory Coalescing

- Coalesced access: consecutive threads access consecutive memory locations



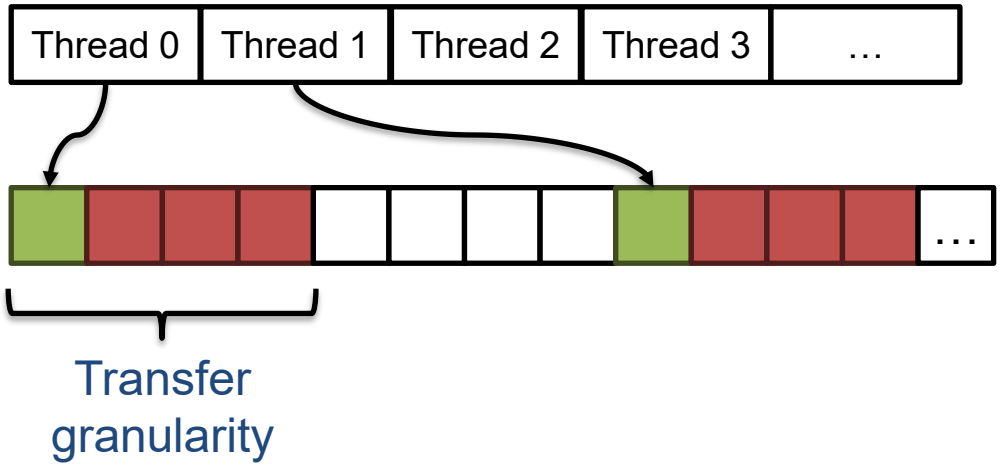
Memory Coalescing

- Uncoalesced access: additional, unused data has to be transferred
- Granularity depends on GPU and on memory space (DRAM, L2, L1, …)



Memory Coalescing

- Uncoalesced access: additional, unused data has to be transferred
- Granularity depends on GPU and on memory space (DRAM, L2, L1, …)



NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Case Study

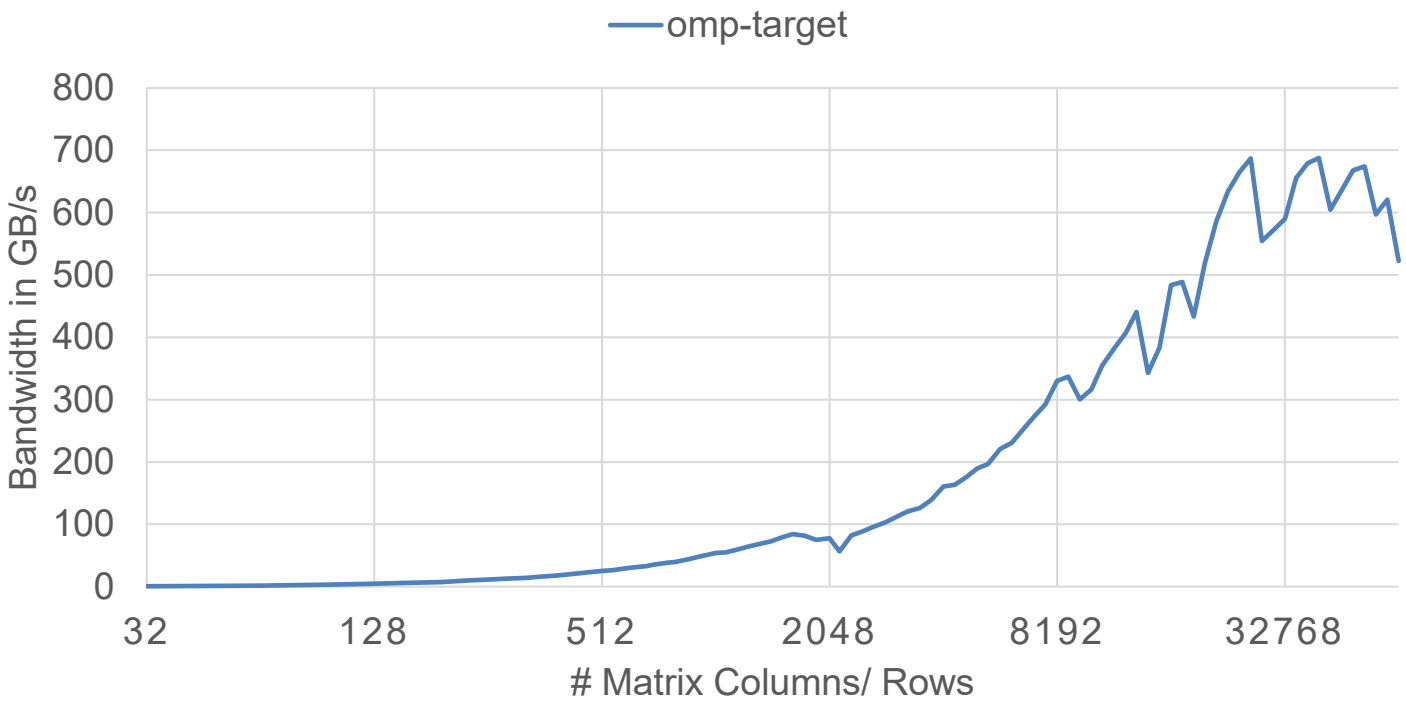
Dense Matrix Vector Multiplication

Case Study dMVM

- Code review
- All results are obtained on a single A100 80GB GPU in Alex
- Source code is available at https://github.com/SebastianKuckuk/apex/tree/main/src/teaching/dmvp

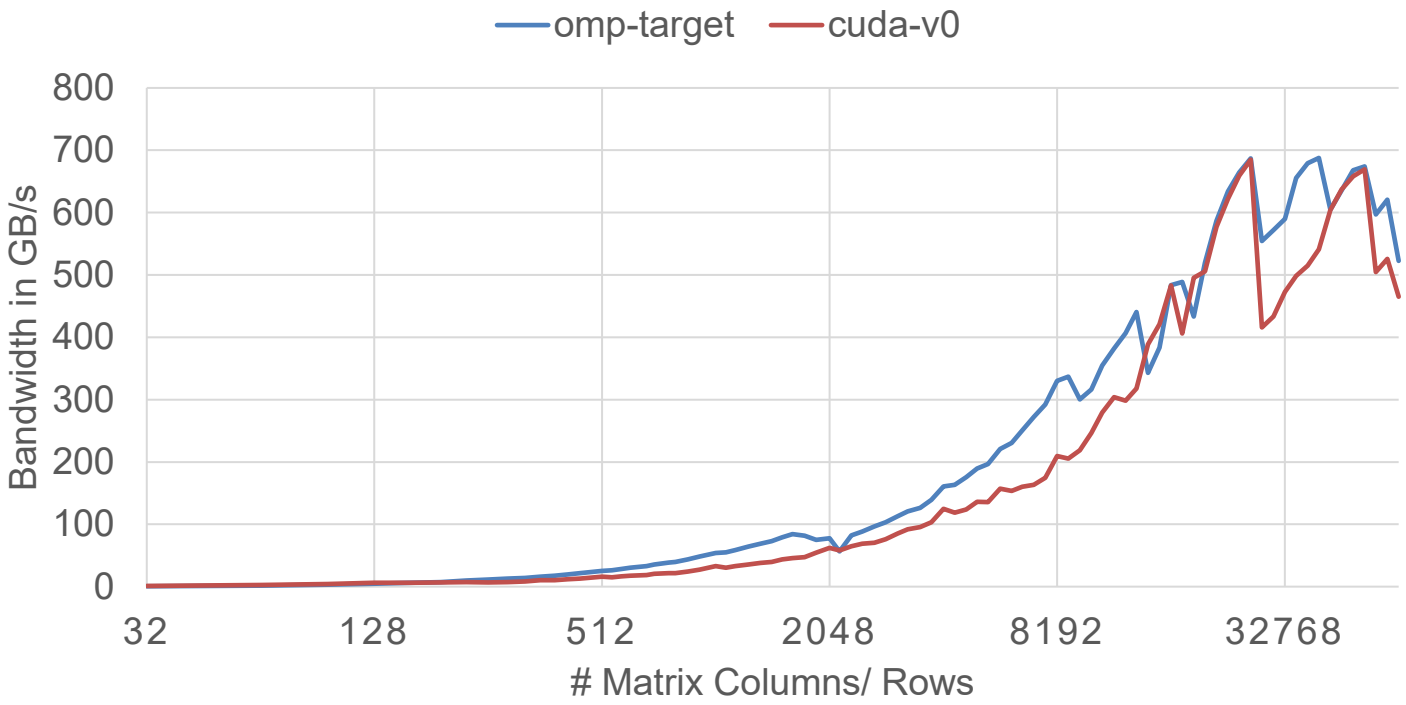
Case Study dMVM

- First Attempt: OpenMP target offload version



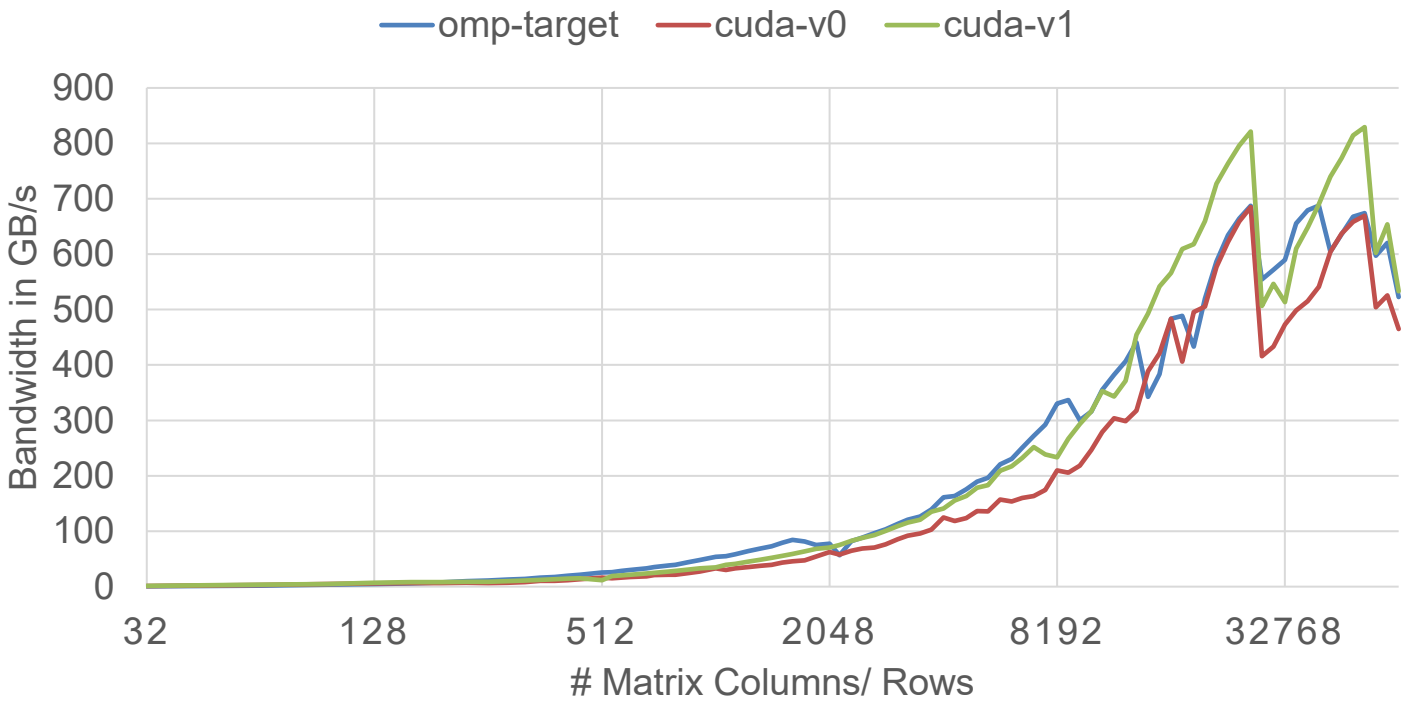
Case Study dMVM

- OpenMP target offload version shows subpar resource utilization
- ➢ Transition to CUDA



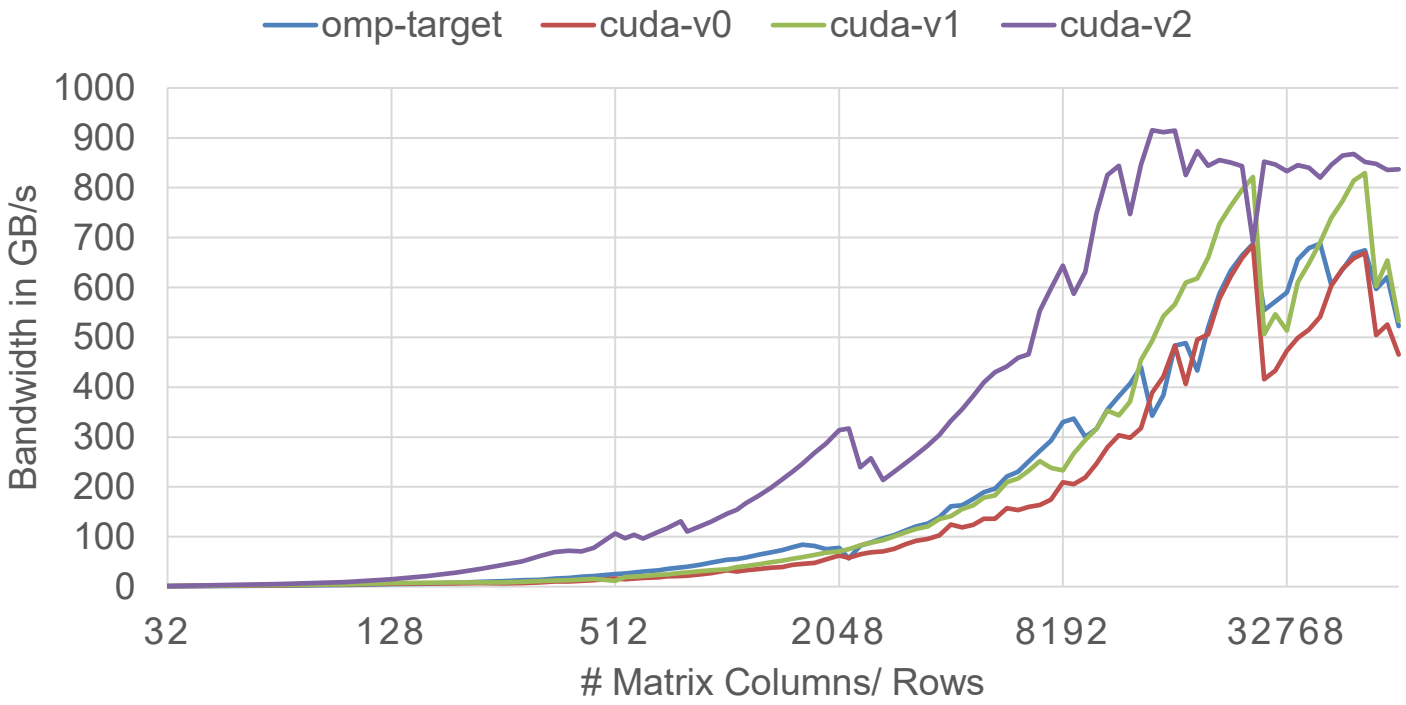
Case Study dMVM

- CUDA v0 frequently reads from/ writes to global memory (may be cached)
- ➢ Use local variable to minimize traffic



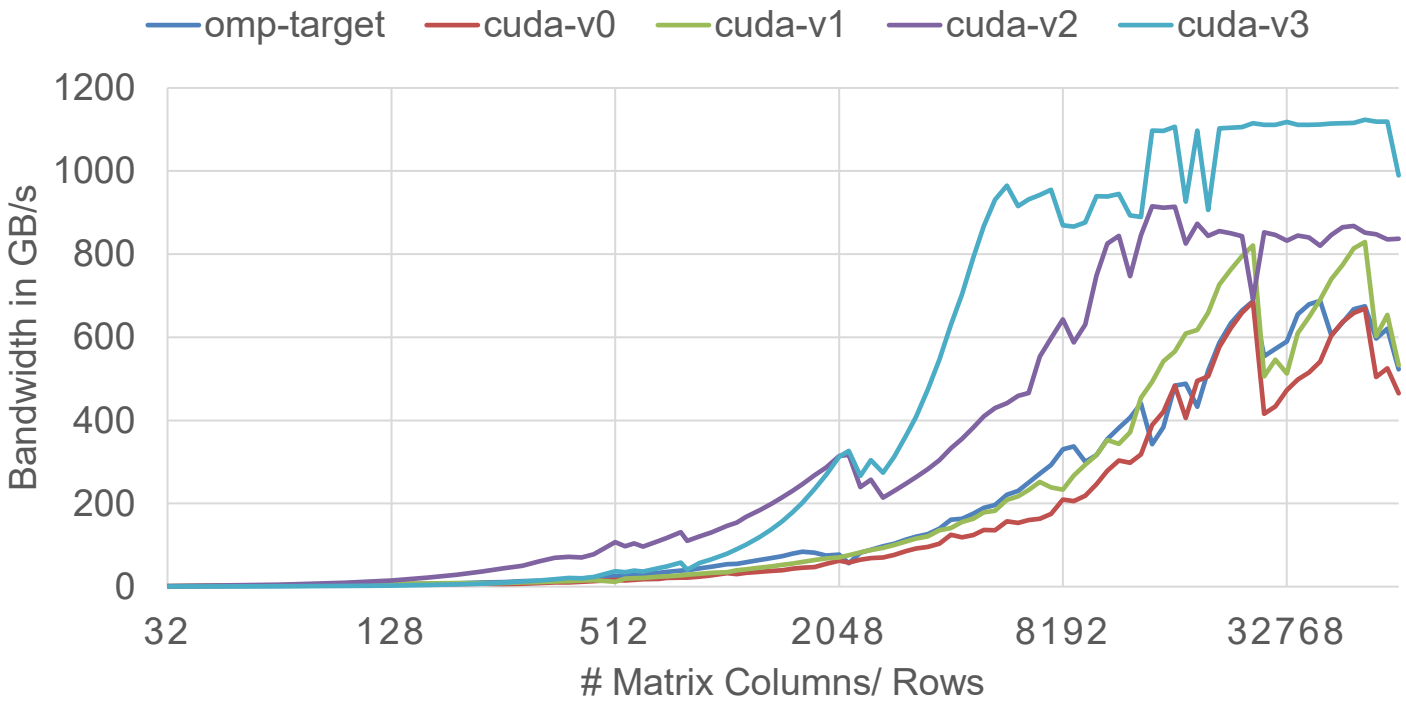
Case Study dMVM

- CUDA v0/1 suffer from a small number of threads
- ➢ Switch execution configuration to 2D and use atomics to avoid data races



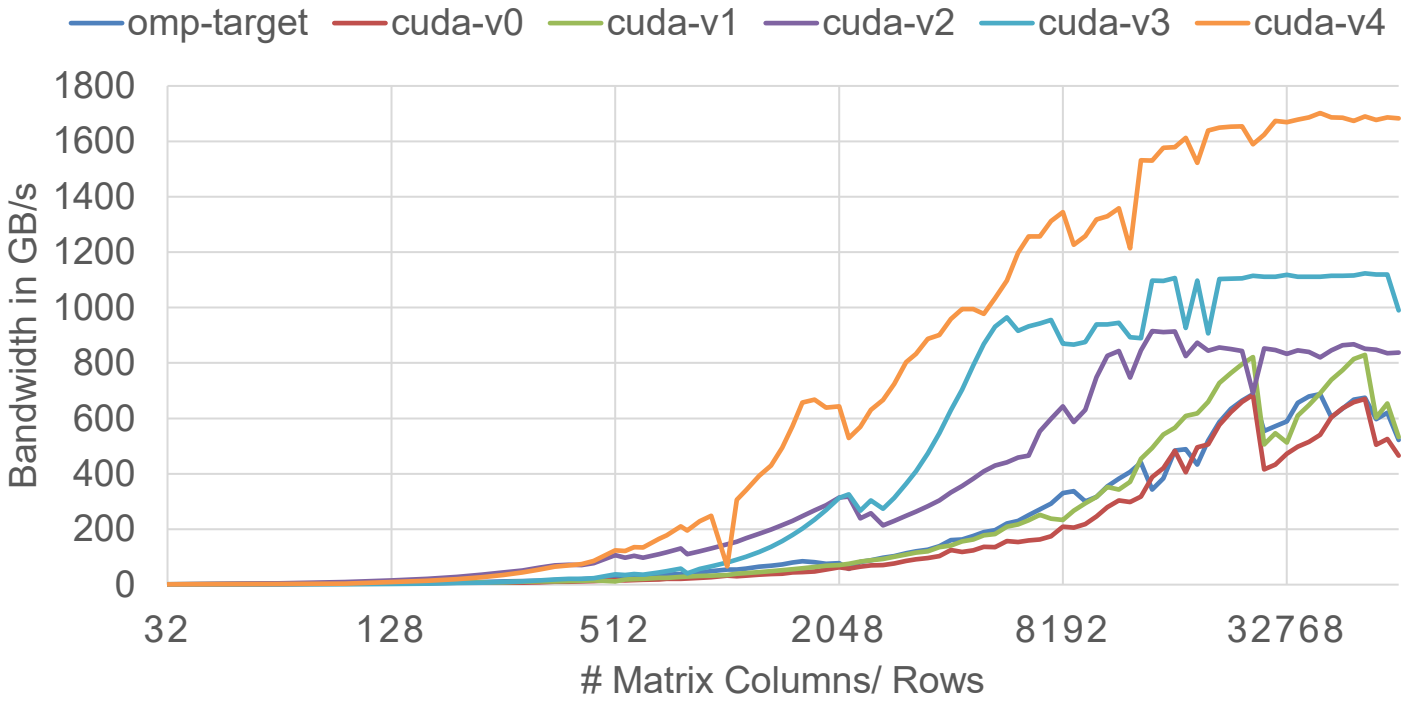
Case Study dMVM

- CUDA v2 experiences atomic congestion and high memory traffic
- ➢ Compute partial results and add grid-stride loops



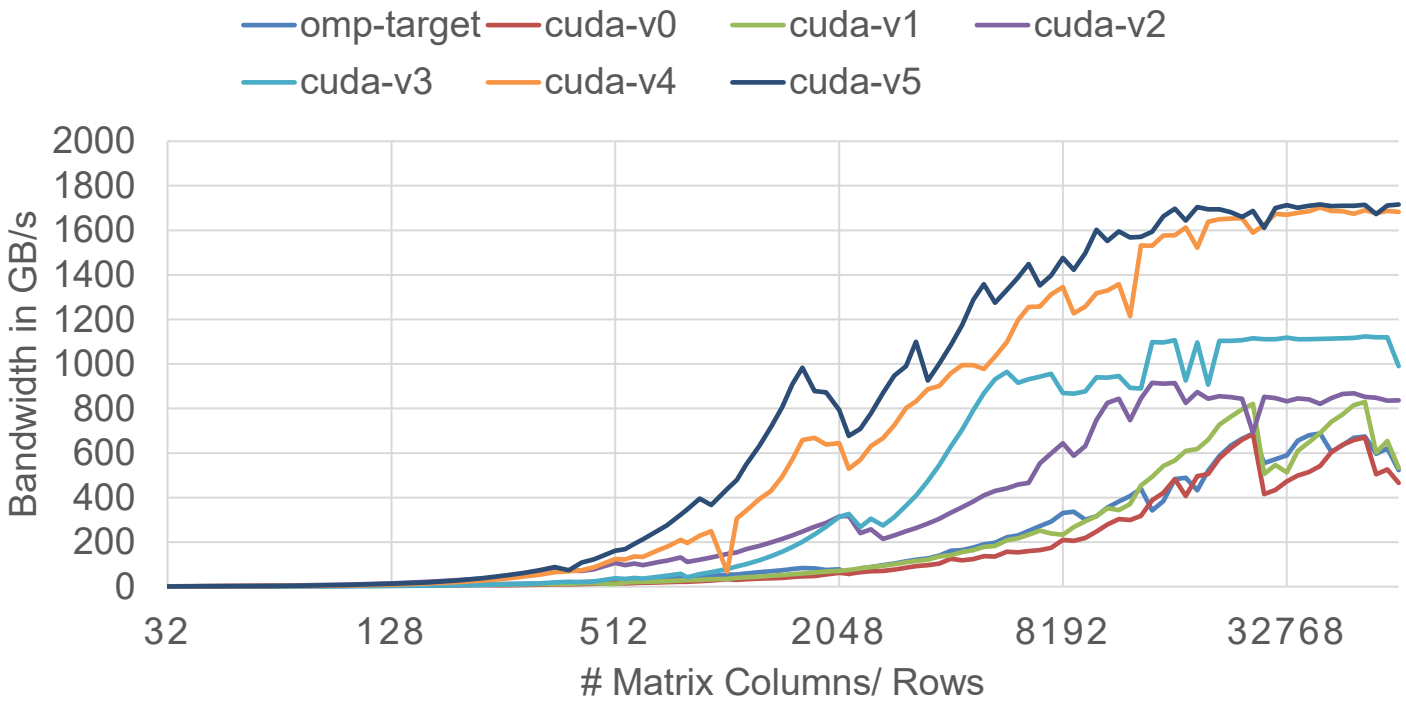
Case Study dMVM

- CUDA v3 writes coalesced but reads uncoalesced
- ➢ Transpose execution configuration



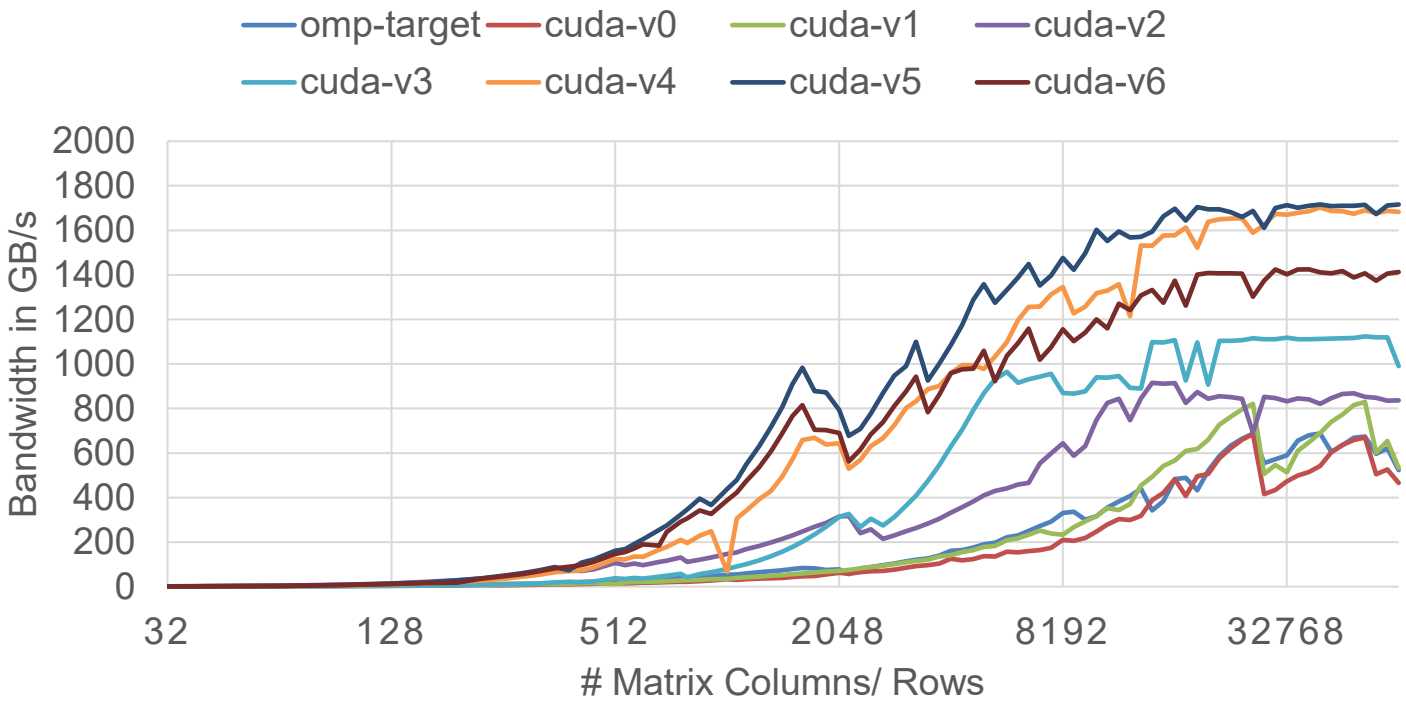
Case Study dMVM

- CUDA v4 doesn't aggregate results within CUDA blocks
- ➢ Use shared memory to buffer and aggregate output data



Case Study dMVM

- CUDA v5 doesn't buffer and share input data
- ➢ Use shared memory to buffer input data



Case Study dMVM

- CUDA v6 is slower (and becomes quite complex/ hard to maintain & tune)
- ➢ Use library implementation (cuBLAS) instead