

HANDS-ON — MEMORY HIERARCHIES IN CPU/GPU ARCHITECTURES

Siegfried Höfinger

ASC Research Center, TU Wien

October 20, 2025

→ <https://tinyurl.com/cudafordummies/i/ho2/notes-ho2.pdf>

Exercise

- Q1)** *For a dummy kernel that does nothing else than reading the content of two arrays, $a[]$ and $b[]$, then adding together element by element and storing the results into a third array, $c[]$, determine the bandwidth with the help of 'nsys nvprof' if we make use of `cudaMallocManaged()` and consider arrays of size 1 GB all throughout.*

- A1)**
- i) *Examine the below sample program and adjust the dimension of the arrays in case,
vi `./unified_memory_example_2.cu`*
 - ii) *Compile and run it via the profiling toolchain
`nvcc unified_memory_example_2.cu`
`nsys nvprof ./a.out`
and read out the time spent in `KrnlDmmyCalc()` $\approx 87946553 \text{ ns} = 0.087946553 \text{ s}$*
 - iii) *Calculate the bandwidth like,
 $3 * 1 \text{ GB} / 0.087946553 \text{ s} = 34.1 \text{ GB/s}$
where the 3 stems from the two read and the one write operation (of arrays `x[]`, `y[]` and `z[]`).*
 - iv) *Have it confirmed by the visual profiler, `nsys-ui`*

Exercise

Q2) *Considering the previous results, can we get closer to the theoretical memory bandwidth of 1555 GB/s if we call the compute kernel repeatedly within a loop over 100 iterations ? How would page faults change then and what else could we do to maximize bandwidth ?*

- A2)**
- i) *Yes, we can do better ! Get the below sample program, edit it and make sure that we really call the calculational kernel within a loop over 100 iterations,*
vi ./unified_memory_example_3.cu
 - ii) *Again, compile it, run it, profile it and compute the obtained bandwidth from the profile(approximately 930 GB/s);*
 - iii) *The number of page faults will most likely have reduced now. Memory prefetching or usage of managed global device memory could further increase the bandwidth;*
 - iv) *Have it confirmed by the visual profiler, nsys-ui*