Hands-On — Memory Hierarchies in CPU/GPU Architectures

Siegfried Höfinger

VSC Research Center, TU Wien

December 3, 2020

→ https://tinyurl.com/cuda4dummies/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 nvprof if we make use of cudaMallocManaged() and consider arrays of size 1 GB all throughout.*

15 min

CUDA managed unified memory cont.

A1)

i) *Examine the below sample program and adjust the dimension of the arrays in case,*
vi ./unified_memory_example_2.cu

ii) *Similar to the earlier exercise, put all the build/compile/run steps into a submit script (or check out the provided template) and submit it to some GPU node (don't forget about prefixing it with nvprof...),*
vi ./slrm.sbmt.q1.scrpt
sbatch ./slrm.sbmt.q1.scrpt

iii) *Confirm proper job termination and inspect results in the slurm-*.out file, then compute the bandwidth (approximately 15 GB/s)*

→ `https://tinyurl.com/cuda4dummies/i/l2/unified_memory_example_2.cu`

Exercise

Q2) *Considering the previous results, can we get closer to the theoretical memory bandwidth of 320 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 ?*

15 min

A2)

i) *Yes, we can do better ! Get the below sample program, edit it and make sure that we really loop over 100 iterations,*
vi ./unified_memory_example_3.cu

ii) *Set up a corresponding submit script, run the job and again compute the bandwidth from the resulting slurm-*.out file (approximately 220 GB/s);*
vi ./slrm.sbmt.q2.scrpt
sbatch ./slrm.sbmt.q2.scrpt

iii) *The number of page faults hasn't changed much. Memory prefetching or usage of managed global device memory could further increase the effective bandwidth;*

→ `https://tinyurl.com/cuda4dummies/i/l2/unified_memory_example_3.cu`