Affinity Subcommittee Report

- Memory Management
- Data-to-Device Affinity
- Taskloop Extensions

Christian Terboven

Co-lead: Jannis Klinkenberg

Memory Management (since OpenMP 5.0)

- Did you know that you can …
  - → … allocate in high-bandwidth memory?

```
#include <omp.h>
double *x = omp_alloc(N * size_of(double), omp_high_bw_mem_alloc));
```

- Recent work:
  - → New allocator traits for finer placement control
    - → `partition`: partitioning of allocated memory over storage resources: `environment`, `nearest`, `blocked`, `interleaved`
    - → `part_size`: specifies the size of parts allocated over storage resources
  - → Allow upper bound and stride for OMP_PLACES
    - → Examples: `OMP_PLACES=cores(4)` or `OMP_PLACES=ll_caches(1:2)`
  - → Unify allocator and target memory runtime routines
    - → Capability to allocate device memory with OpenMP allocators: new routines returning target memory spaces
    - → Memory space containing storage resources accessible by all devices as requested

Thread-to-Device Affinity (OpenMP 6.0) / 1

- **Idea:** Find devices that are close to the current thread

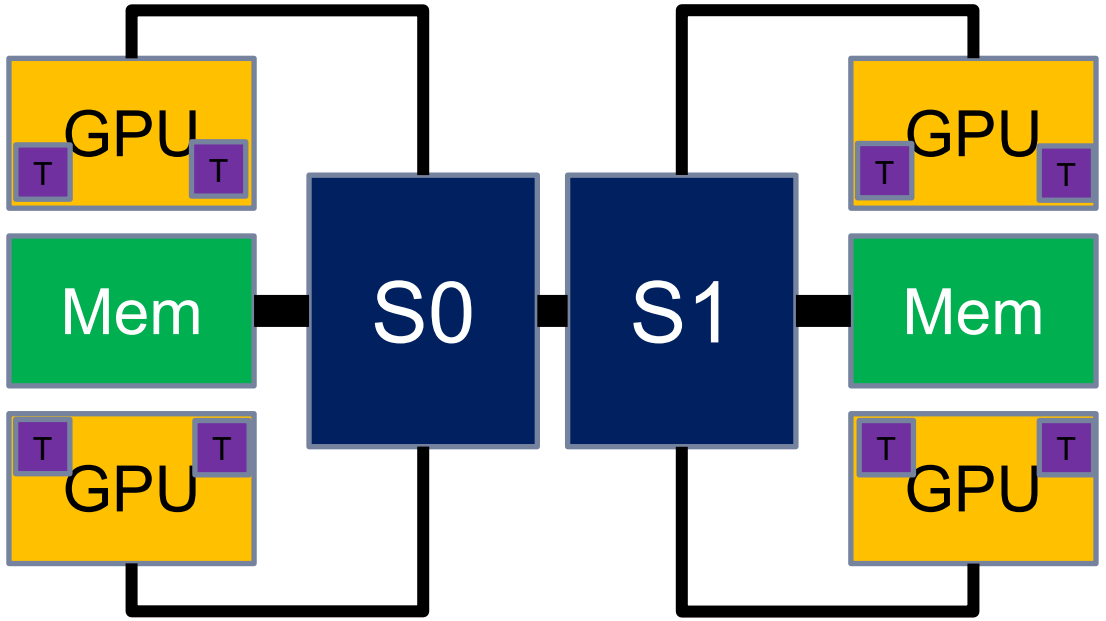
1. **Find devices that are close to the current thread**

```
int n=20;              // desired number of devices
int n_dev_found;       // actual number of devices
int dev_ids[n];
n_dev_found = omp_get_devices_in_order
              (n, dev_ids, <trait_lowest_distance>);

#pragma omp target device(dev_ids[0])
...
#pragma omp target device(dev_ids[n_dev_found-1])
```

2. Use devices that are close to data used in target

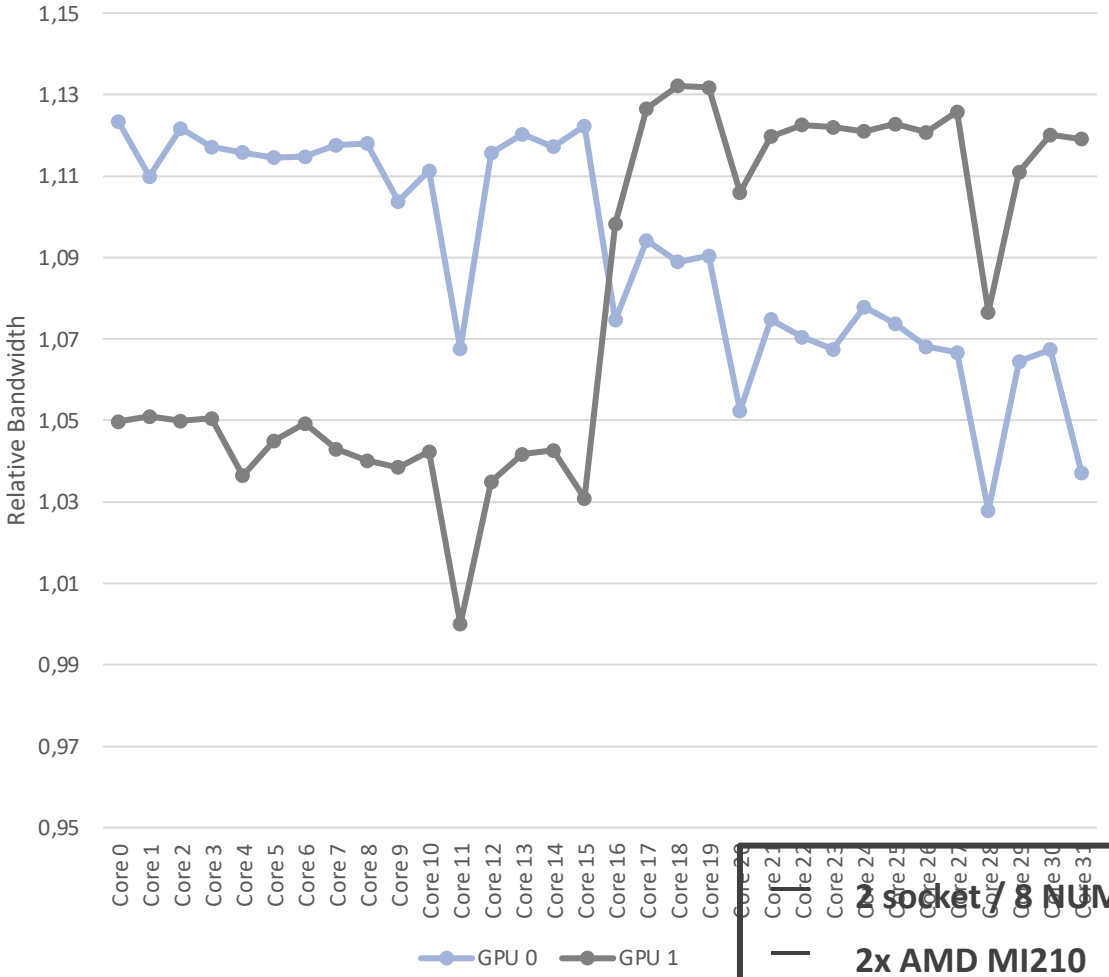
```
#pragma omp task affinity(data[start:len])
{
    #pragma omp target \
        map(tofrom: data[start:len]) \
        device_affinity(data[start:len])
    {
        // content of the target task
    } }
```

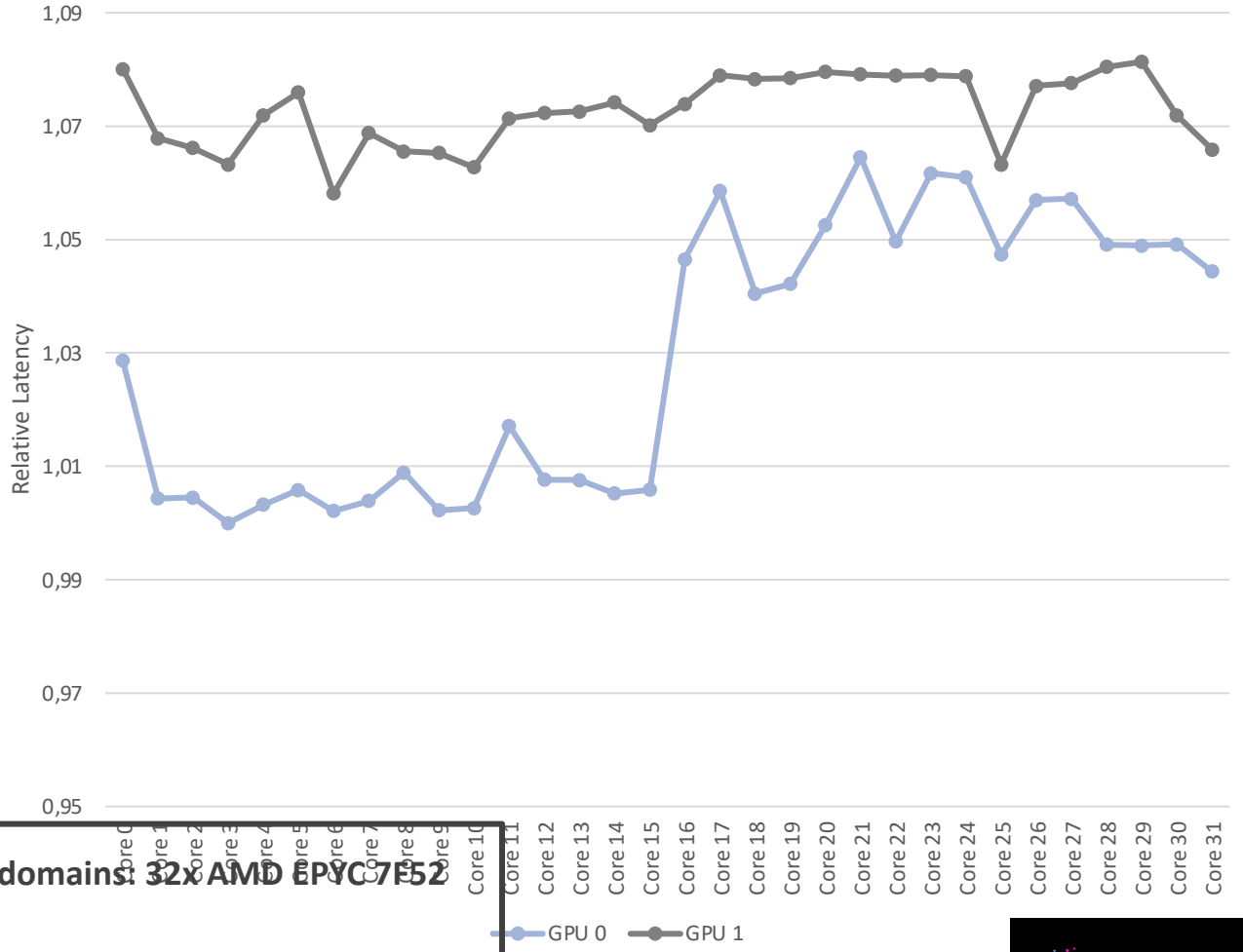


Thread-to-Device Affinity (OpenMP 6.0) / 2

- **Question:** Does it matter?

Here: measurement on AMD MI210





2 socket / 8 NUMA domains: 32x AMD EPYC 7F52

2x AMD MI210

- GPU 0 connected to NUMA domain 2
- GPU 1 connected to NUMA domain 7

Taskloop Extensions

- Motivation (excerpt)
  - → Support the `depend` clause in the taskloop construct (#2072)
  - → Support `affinity` clause for taskloop (#2142)
  - → Support in OpenMP for Parallelization across Multiple OpenMP Devices (#2636)
- Sketch of the idea (the concrete syntax is changing faster than my slides)
  - → One mechanism to record the iteration space decomposition
  - → Another mechanism to restore/reproduce the iteration space, or access it manually

```
omp_chunks_t myChunks;

#pragma omp taskloop depend(inout:v[omp_chunk[0].lower;omp_chunk[0].upper]) save(myChunks) nogroup
for (int i = 0; i < N; ++i) { v[i] += ...; }

for (int ch = 0; ch < myChunks.num_chunks; ch++){
   #pragma omp task depend(inout: v[myChunks.chunks[ch].lower;myChunks.chunks[ch].upper])
   { ... for (int ii = myChunks[ch].lower; ii < myChunks[ch].upper; ii++){ v[i] += ...; }... }
}
```