TÜBİTAK

Sabancı Üniversitesi

EURO^2

Offloading Computation to a GPU with OpenMP

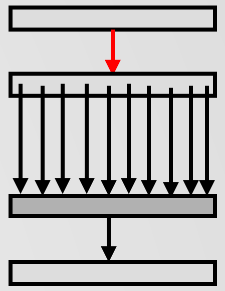
Kamer Kaya, Sabancı University

ncc@ulakbim.gov.tr

Offloading to GPU – More detail
Work sharing

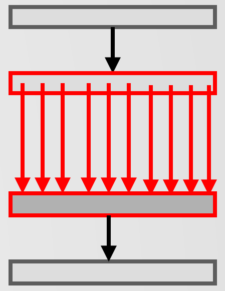
#pragma omp teams num_teams (K)

- Coarse grain parallelism
- Generates multiple, single-thread teams,
- Teams map to SMs in HW (similar to blocks in CUDA)
- No synchronization among the teams (similar to blocks in CUDA)



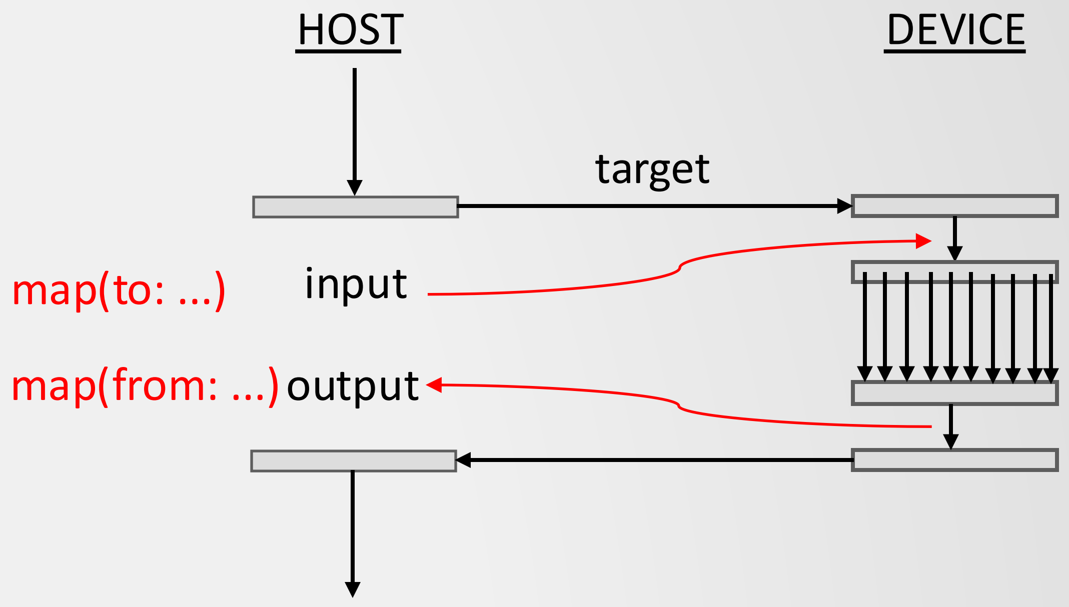
#pragma omp parallel

- Fine grain parallelism
- Generates many threads in a team
- Threads map to CUDA cores in HW (similar to threads in CUDA)
- Synchronization inside a team is possible (similar to block-level sync. in CUDA)



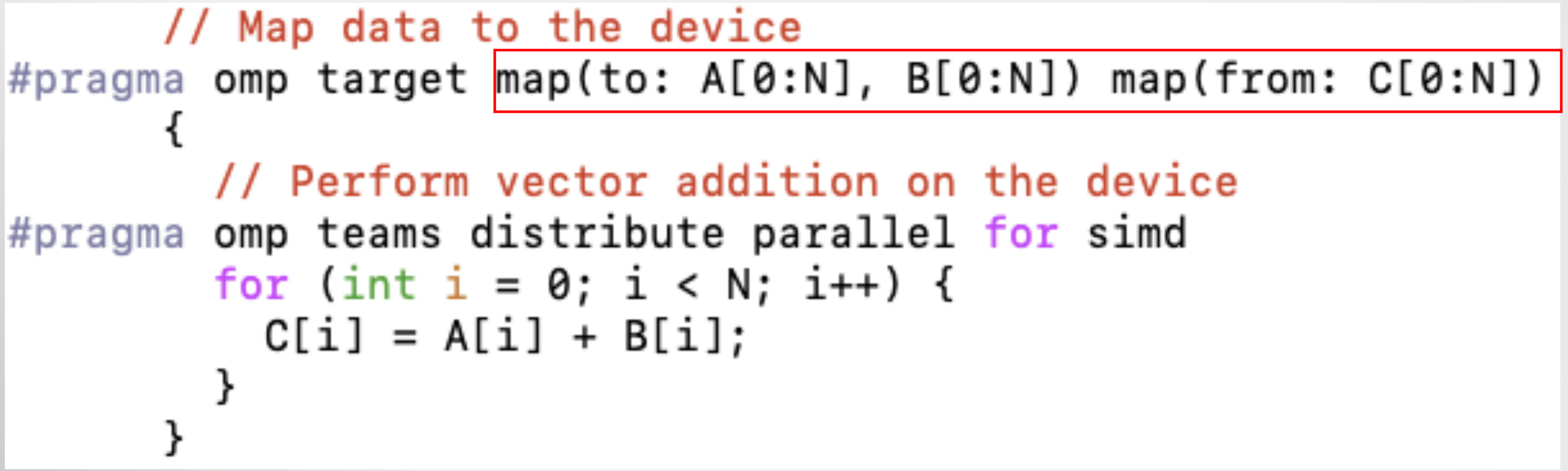
Offloading to GPU – More detail
Data mapping

- Data mapping is a mechanism provided to share variables between the host and device;
  - Memory may or may not be shared; the implementation determines whether data is copied or not.
  - A mapped data represents the device-side counterpart of a host data in the device data environment, similar to shared data.



Offloading to GPU – More detail
Data mapping

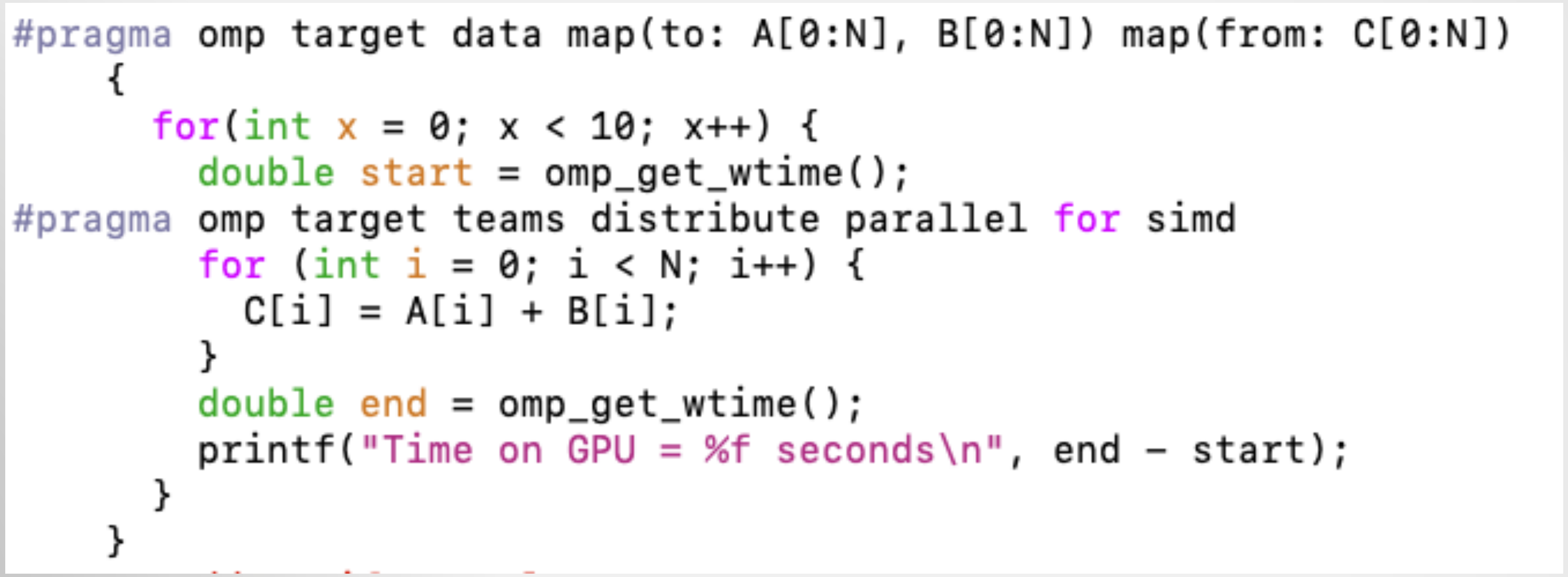
- Arrays can be mapped by using [,] syntax as shown below for a vector addition.



```
      // Map data to the device
#pragma omp target map(to: A[0:N], B[0:N]) map(from: C[0:N])
      {
        // Perform vector addition on the device
#pragma omp teams distribute parallel for simd
        for (int i = 0; i < N; i++) {
          C[i] = A[i] + B[i];
        }
      }
```

Offloading to GPU – More detail
Data mapping

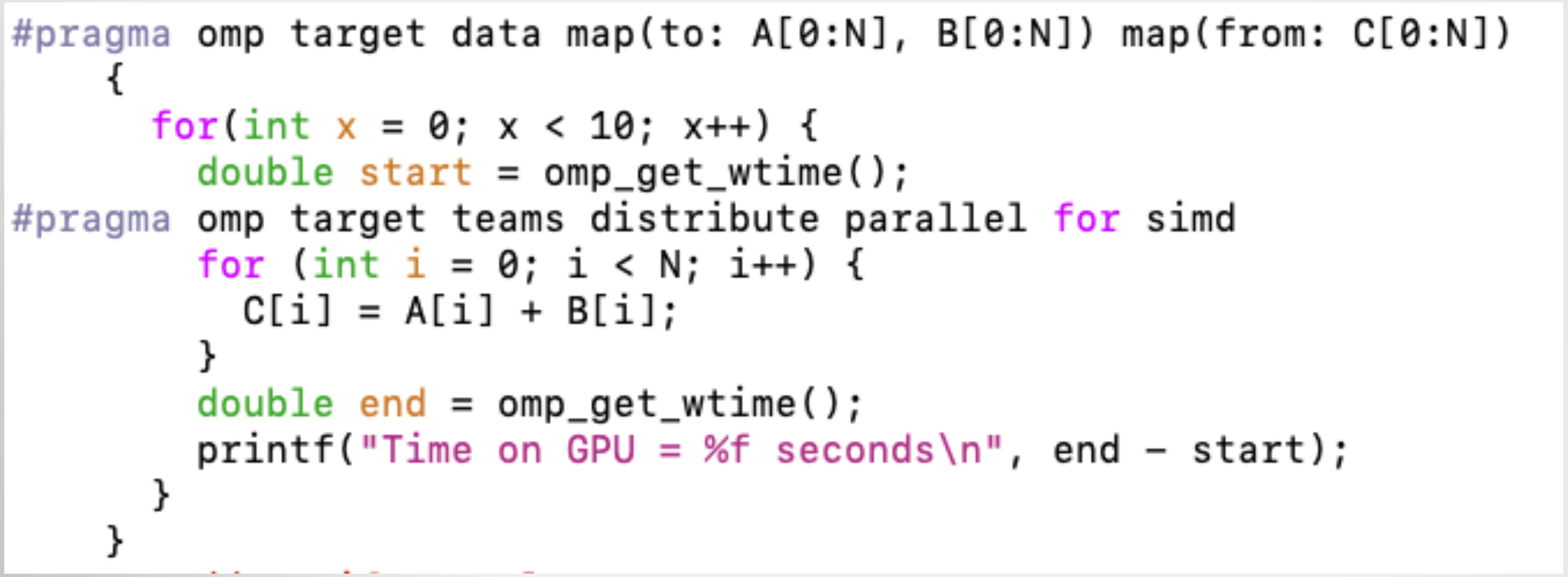
- Data transfers are slow, we need to minimize as much as possible.
- We can use **target data** to reuse the data on the GPU.



```
#pragma omp target data map(to: A[0:N], B[0:N]) map(from: C[0:N])
    {
      for(int x = 0; x < 10; x++) {
        double start = omp_get_wtime();
#pragma omp target teams distribute parallel for simd
        for (int i = 0; i < N; i++) {
          C[i] = A[i] + B[i];
        }
        double end = omp_get_wtime();
        printf("Time on GPU = %f seconds\n", end - start);
      }
    }
```

Offloading to GPU – More detail
Data mapping

- Data transfers are slow, we need to minimize as much as possible.
- We can use **target data** to reuse the data on the GPU.
    - and use **target update** when required.



```
#pragma omp target data map(to: A[0:N], B[0:N]) map(from: C[0:N])
    {
      for(int x = 0; x < 10; x++) {
        double start = omp_get_wtime();
#pragma omp target teams distribute parallel for simd
        for (int i = 0; i < N; i++) {
          C[i] = A[i] + B[i];
        }
        double end = omp_get_wtime();
        printf("Time on GPU = %f seconds\n", end - start);
      }
    }
```

Offloading to GPU – More detail
Work sharing – simd

simd

```
"Requested Global Load Throughput", 118.694765MB/s, 119.051373MB/s, 118.872715MB/s
"Requested Global Store Throughput", 191.918411MB/s, 192.680808MB/s, 192.298853MB/s
"Global Load Throughput", 830.751312MB/s, 852.950066MB/s, 841.872691MB/s
"Global Store Throughput", 1.133424GB/s, 1.175496GB/s, 1.154501GB/s
```

no simd

```
"Requested Global Load Throughput", 4.358355MB/s, 4.361329MB/s, 4.359840MB/s
"Requested Global Store Throughput", 6.945406MB/s, 6.953836MB/s, 6.949618MB/s
"Global Load Throughput", 32.433474MB/s, 32.759454MB/s, 32.596562MB/s
"Global Store Throughput", 42.796398MB/s, 42.872280MB/s, 42.834362MB/s
```

EURO²

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia