Introduction to OpenMP

Dr. Christian Terboven

NUMA Architectures

Dr. Christian Terboven

Introduction to OpenMP

Non-Uniform Memory Architecture

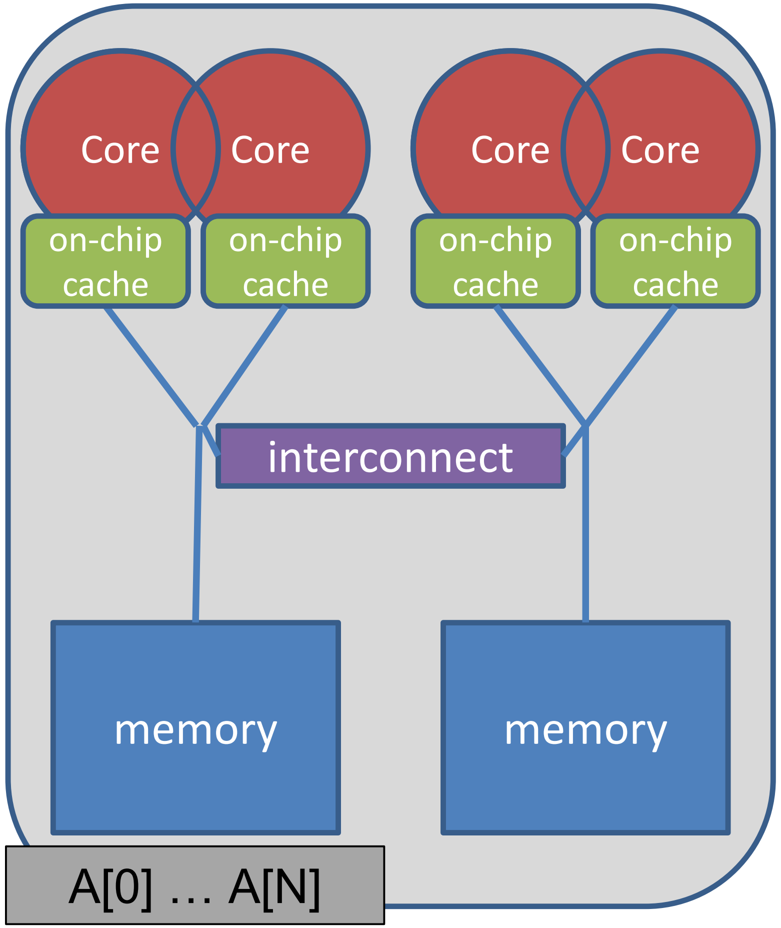
HPC.NRW

How To Distribute The Data ?

```
double* A;
A = (double*)
    malloc(N * sizeof(double));



for (int i = 0; i < N; i++) {
   A[i] = 0.0;
}
```



About Data Distribution

- Important aspect on cc-NUMA systems
  - If not optimal, longer memory access times and hotspots

- OpenMP does not provide explicit support for cc-NUMA on first sight

- Placement comes from the Operating System
  - This is therefore Operating System dependent
  - OpenMP 5.0 introduced Memory Management to provide fine-grained control

- Windows, Linux and Solaris all use the “First Touch” placement policy by default
  - May be possible to override default (check the docs)

Non-Uniform Memory Architecture

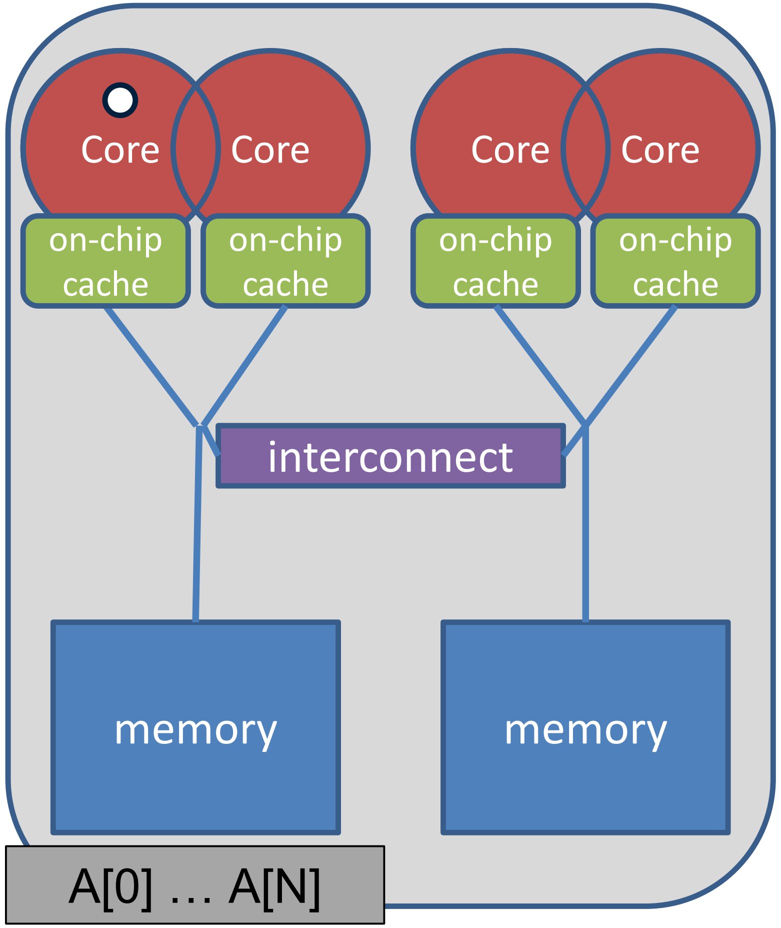
HPC.NRW

– Serial code: all array elements are allocated in the memory of the NUMA node containing the core executing this thread

```
double* A;
A = (double*)
    malloc(N * sizeof(double));



for (int i = 0; i < N; i++) {
   A[i] = 0.0;
}
```



Non-Uniform Memory Architecture

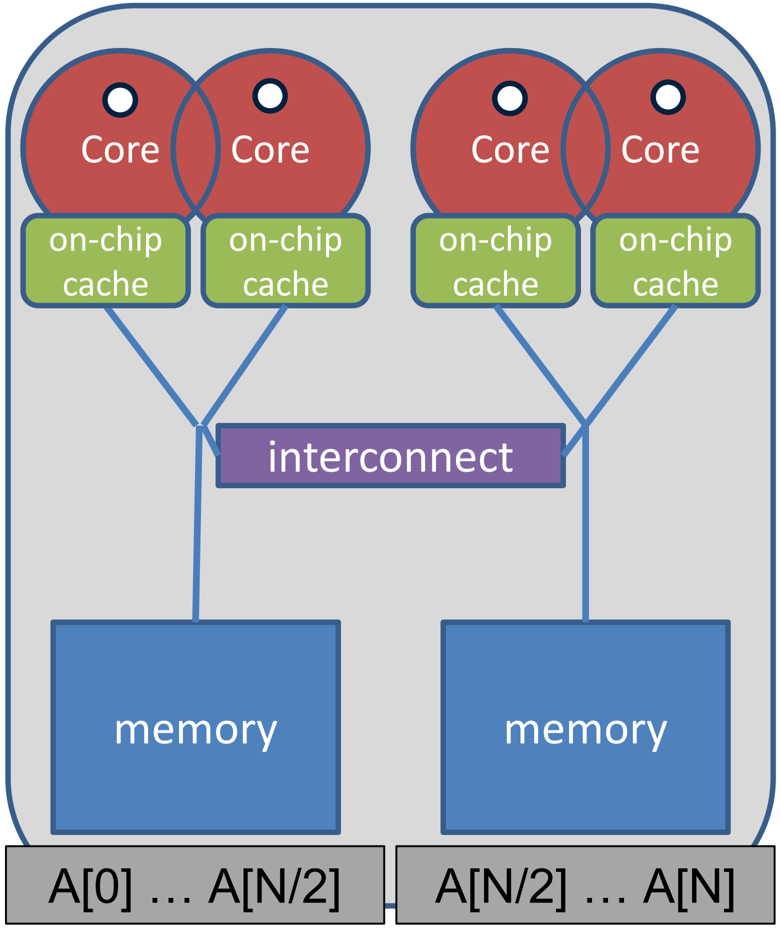
HPC.NRW

– First Touch w/ parallel code: all array elements are allocated in the memory of the NUMA node containing the core executing the thread initializing the respective partition

```
double* A;
A = (double*)
    malloc(N * sizeof(double));

omp_set_num_threads(4);

#pragma omp parallel for
for (int i = 0; i < N; i++) {
   A[i] = 0.0;
}
```



Get Info on the System Topology

HPC.NRW

- Before you design a strategy for thread binding, you should have a basic understanding of the system topology. Please use one of the following options on a target machine:

  - Intel MPI‘s `cpuinfo` tool
    - `module switch openmpi intelmpi`
    - `cpuinfo`
    - Delivers information about the number of sockets (= packages) and the mapping of processor ids used by the operating system to cpu cores.

  - hwlocs‘tools
    - `lstopo` (command line: `hwloc-ls`)
    - Displays a graphical representation of the system topology, separated into NUMA nodes, along with the mapping of processor ids used by the operating system to cpu cores and additional info on caches.

Decide for Binding Strategy

HPC.NRW

- Selecting the „right“ binding strategy depends not only on the topology, but also on the characteristics of your application.

  - Putting threads far apart, i.e. on different sockets
    - May improve the aggregated memory bandwidth available to your application
    - May improve the combined cache size available to your application
    - May decrease performance of synchronization constructs

  - Putting threads close together, i.e. on two adjacent cores which possibly shared some caches
    - May improve performance of synchronization constructs
    - May decrease the available memory bandwidth and cache size

- If you are unsure, just try a few options and then select the best one.

OpenMP 4.0: Places + Binding Policies (1/2)

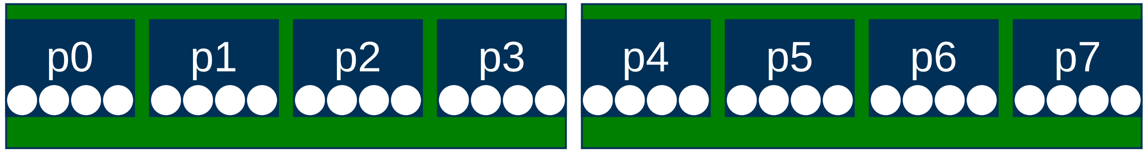
- Define OpenMP Places
  - set of OpenMP threads running on one or more processors
  - can be defined by the user, i.e. `OMP_PLACES=cores`

- Define a set of OpenMP Thread Affinity Policies
  - SPREAD: spread OpenMP threads evenly among the places
  - CLOSE: pack OpenMP threads near master thread
  - MASTER: collocate OpenMP thread with master thread

- Goals
  - user has a way to specify where to execute OpenMP threads for
  - locality between OpenMP threads / less false sharing / memory bandwidth

Places

– Assume the following machine:



– 2 sockets, 4 cores per socket, 4 hyper-threads per core

– Abstract names for OMP_PLACES:
  – threads: Each place corresponds to a single hardware thread on the target machine.
  – cores: Each place corresponds to a single core (having one or more hardware threads) on the target machine.
  – sockets: Each place corresponds to a single socket (consisting of one or more cores) on the target machine.
  – ll_caches (5.1): Each place corresponds to a set of cores that share the last level cache.
  – numa_domains (5.1): Each places corresponds to a set of cores for which their closest memory is: the same memory; and at a similar distance from the cores.

OpenMP 4.0: Places + Binding Policies (2/2)

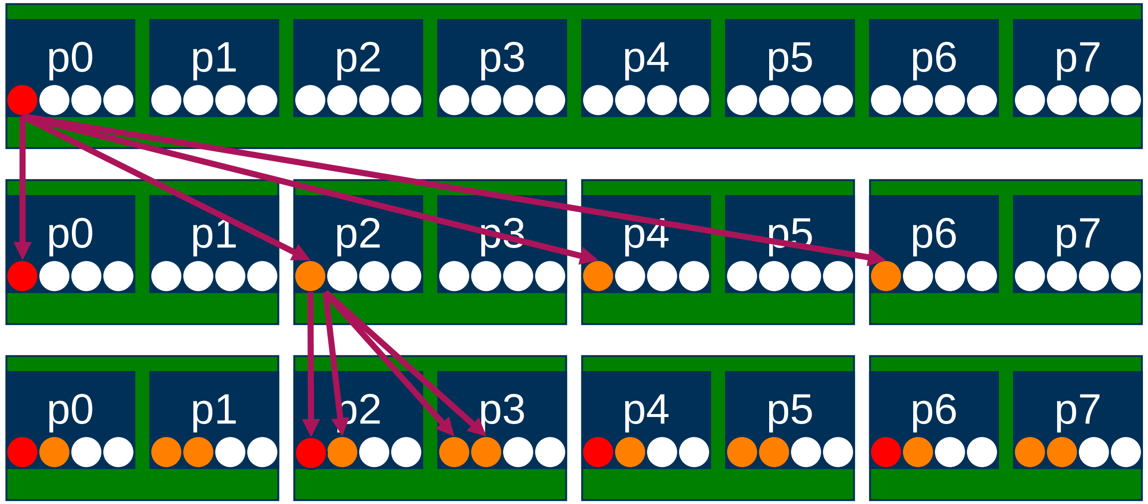
HPC.NRW

- Example's Objective:
  - separate cores for outer loop and near cores for inner loop
- Outer Parallel Region: proc_bind(spread), Inner: proc_bind(close)
  - spread creates partition, compact binds threads within respective partition

```
OMP_PLACES=(0,1,2,3), (4,5,6,7), ... = (0-3):8:4    = cores
```

```
#pragma omp parallel proc_bind(spread) num_threads(4)
#pragma omp parallel proc_bind(close) num_threads(4)
```

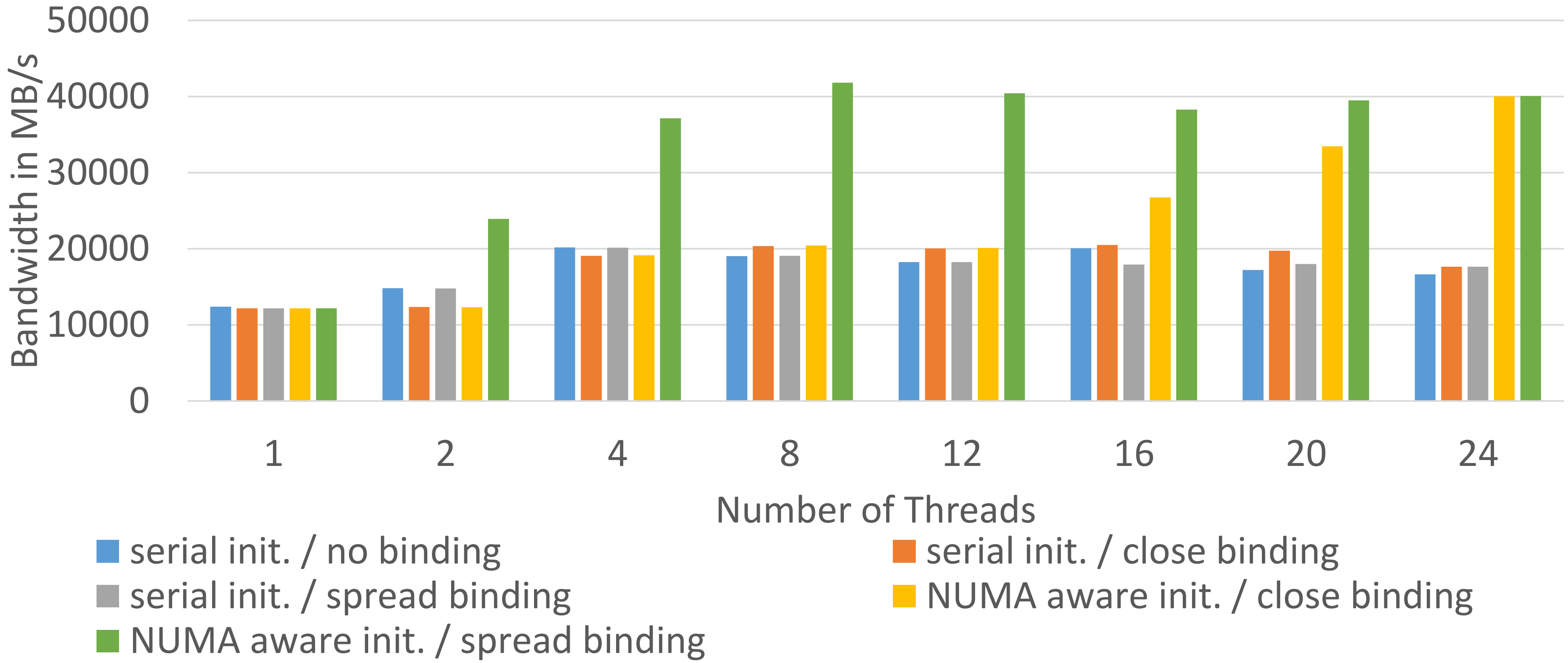
- Example
  - initial
  - spread 4
  - close 4



Serial vs. Parallel Initialization

HPC.NRW

– Performance of OpenMP-parallel STREAM vector assignment measured on 2-socket Intel® Xeon® X5675 („Westmere“) using Intel® Composer XE 2013 compiler with different thread binding options:



Questions?