Performance Considerations on NUMA Machines

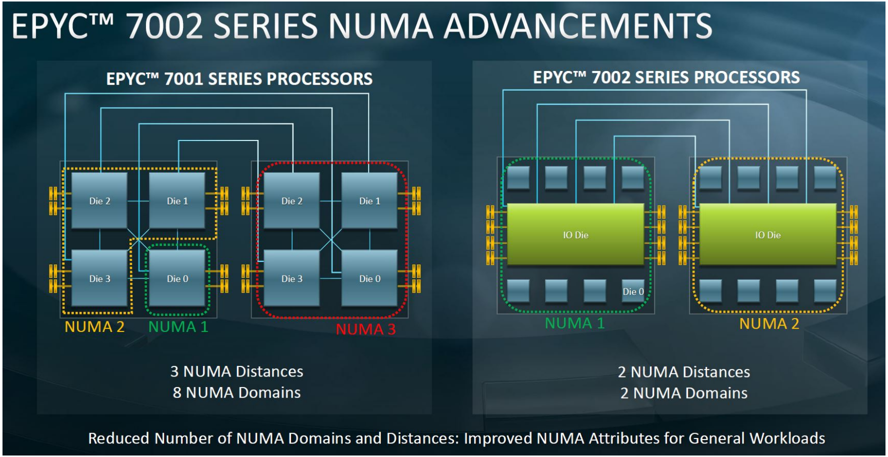
Basics of Parallel Computing

Assoc.Prof. Dr. Sascha Hunold

TU Wien

NUMA Performance in Practice

NUMA Examples / AMD EPYC / Intel Xeon I



source: `https://www.servethehome.com/wp-content/uploads/2019/08/AMD-EPYC-7002-Architecture-NUMA-Reduction-to-104ns-Close-201ns-Far.jpg`

NUMA Examples / AMD EPYC / Intel Xeon II

- our machine nebula

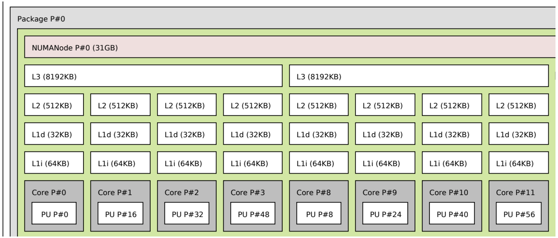
```
hunold@nebulac:~$ lscpu |grep "Model name"
Model name:          AMD EPYC 7551 32-Core Processor
```

- `https://www.amd.com/de/products/cpu/amd-epyc-7551`
  - Product line: AMD EPYC 7001 Series
- `hwloc` can help to reveal processor details and topology

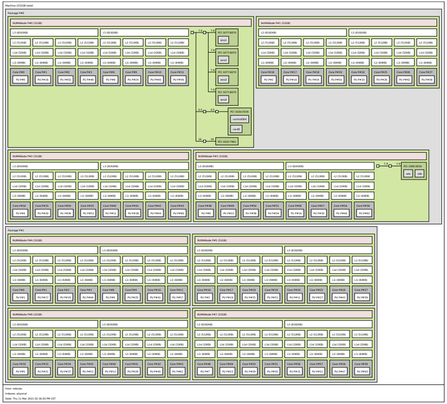
```
hunold@nebulac:~$ hwloc-ls - --of pdf > nebulac_node.pdf
```

NUMA Examples / AMD EPYC / Intel Xeon III

- one NUMA node in detail
- notice the core numbering!
  - cores 0, 16, 32, 48 share one L3 cache (important)



in total: 8 NUMA nodes (2×4)



NUMA Examples / AMD EPYC / Intel Xeon IV

Experiment

- we check the bandwidth (how many bytes we can read and write from/to memory)
  - we say bandwidth but actually mean throughput
  - bandwidth is theoretical, throughput is a practical measure
- we use 2 threads and nebula

config 1

- we pin both threads to one NUMA node

config 2

- we pin both threads to different NUMA nodes

hypothesis

- config 2 gives more bandwidth as we can leverage more memory controllers

config 1 (same NUMA node)

```
hunold@nebulac:~/tmp/stream$ numactl --physcpubind=0,16 ./stream_omp
-------------------------------------------------------------
Number of Threads requested = 2
Number of Threads counted = 2
-------------------------------------------------------------
Function     Best Rate MB/s  Avg time     Min time     Max time
Copy:           20735.7      0.007731     0.007716     0.007753
Scale:          20562.8      0.007788     0.007781     0.007798
Add:            22720.5      0.010576     0.010563     0.010584
Triad:          22735.9      0.010566     0.010556     0.010578
```

- memory throughput: ~23 GB/s

NUMA Examples / AMD EPYC / Intel Xeon VI

config 2 (different NUMA nodes)

```
hunold@nebulac:~/tmp/stream$ numactl --physcpubind=0,2 ./stream_omp
-------------------------------------------------------------
Number of Threads requested = 2
Number of Threads counted = 2
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           30805.1     0.005221     0.005194     0.005262
Scale:          31658.1     0.005064     0.005054     0.005077
Add:            39435.6     0.006102     0.006086     0.006113
Triad:          38609.7     0.006235     0.006216     0.006257
```

- memory throughput: ~39 GB/s

NUMA Examples / AMD EPYC / Intel Xeon VII

- memory latencies between NUMA nodes

```
hunold@nebulac:~$ numactl -H
available: 8 nodes (0-7)
node 0 cpus: 0 8 16 24 32 40 48 56
node 0 size: 31934 MB
node 0 free: 6940 MB
node 1 cpus: 2 10 18 26 34 42 50 58
node 1 size: 32252 MB
node 1 free: 30238 MB
node 2 cpus: 4 12 20 28 36 44 52 60
node 2 size: 32252 MB
node 2 free: 31221 MB
// .. I removed a few lines
node distances:
node   0   1   2   3   4   5   6   7
  0:  10  16  16  16  28  28  22  28
  1:  16  10  16  16  28  28  28  22
  2:  16  16  10  16  22  28  28  28
  3:  16  16  16  10  28  22  28  28
  4:  28  28  22  28  10  16  16  16
  5:  28  28  28  22  16  10  16  16
  6:  22  28  28  28  16  16  10  16
  7:  28  22  28  28  16  16  16  10
```

NUMA Examples / AMD EPYC / Intel Xeon VIII

- let us look at a hydra node

```
hunold@hydra01:~$ lscpu  | grep "Model name"
Model name:          Intel(R) Xeon(R) Gold 6130F CPU @ 2.10GHz

hunold@hydra01:~$ hwloc-ls - --of pdf > hydra_node.pdf
```

- thus, sockets are NUMA nodes

NUMA Examples / AMD EPYC / Intel Xeon X

- memory latency to another NUMA node is roughly twice higher

```
hunold@hydra01:~$ numactl -H
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
node 0 size: 46803 MB
node 0 free: 42308 MB
node 1 cpus: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
node 1 size: 48360 MB
node 1 free: 46808 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```

Does this inter-core latency matter? I

Experiment

- OSU MPI Microbenchmarks
- 2 processes perform a ping-pong
  - A sends message to B, B sends message back
  - we measure the (mean) time to complete this ping-pong
- we do this on one compute node of hydra

config 1

- processes 1 and 2 are mapped to same socket

config 2

- processes 1 and 2 are mapped to different sockets

hypothesis

- config 2 is slower than config 1 (at least for small messages)
- since physical latency within one socket should be smaller (see slide before)

Does this inter-core latency matter? II

- config 1 (on same socket)

```
srun --nodelist=hydra01 --ntasks-per-node=2 \
-m block:block ./mpi/pt2pt/osu_latency -m 1:128 \
-i 10000

# OSU MPI Latency Test v5.4.1
# Size          Latency (us)
1                       0.41
2                       0.41
4                       0.40
8                       0.41
16                      0.41
32                      0.56
64                      0.55
128                     0.58
```

- config 2 (on different sockets)

```
srun --nodelist=hydra01 --ntasks-per-node=2 \
-m block:cyclic ./mpi/pt2pt/osu_latency -m 1:128 \
-i 10000

# OSU MPI Latency Test v5.4.1
# Size          Latency (us)
1                       0.65
2                       0.65
4                       0.65
8                       0.65
16                      0.65
32                      1.01
64                      1.00
128                     1.01
```

- indeed, config 1 is faster
- take away message: placement of processes and threads (on which core) is important