

# Parallel Programming and Heterogeneous Computing

## Non-Uniform Memory Access

Max Plauth, Sven Köhler, *Felix Eberhardt*, Lukas Wenzel and Andreas Polze

Operating Systems and Middleware Group

# Recap

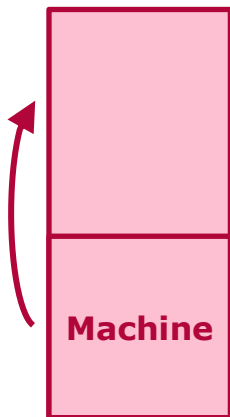
## Optimization Goals

- Decrease **Latency** – process a single workload faster (= **speedup**)
  - Increase **Throughput** – process more workloads in the same time
  - Both are **Performance** metrics
- **Scalability**: make best use of additional resources
    - **Scale Up**: Utilize additional resources on a machine
    - **Scale Out**: Utilize resources on additional machines
- **Cost/Energy Efficiency**:
    - minimize cost/energy requirements for given performance objectives
    - *alternatively: maximize performance for given cost/energy budget*
  - **Utilization**: minimize idle time (=waste) of available resources
  - **Precision-Tradeoffs**: trade performance for precision of results

# Non-Uniform Memory Access

## Context: Scalability

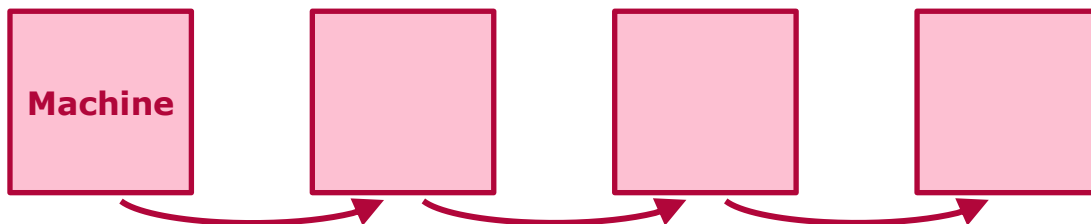
- Two basic approaches to scaling computing hardware:
  - **Scale-Up:** combine more resources (memory or cores) in a tightly coupled system
    - User perceives a single large shared-memory system



# Non-Uniform Memory Access

## Context: Scalability

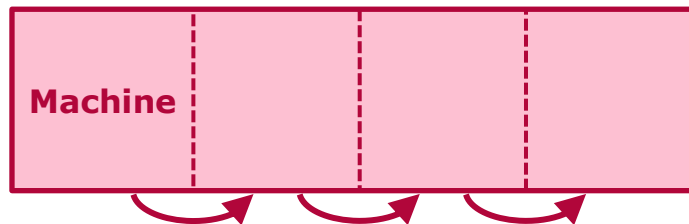
- Two basic approaches to scaling computing hardware:
  - **Scale-Out:** connect more machines in a loosely coupled network
    - User perceives multiple communicating machines in a shared-nothing system



# Non-Uniform Memory Access

## Context: Scalability

- Recent coherent interconnect technologies enable **hybrid** systems with both scale-up and scale-out characteristics:
  - Example: Gen-Z strives to connect an entire datacenter of machines coherently
  - User perceives a shared-memory system, but with the performance characteristics (communication latency and bandwidth) of a shared-nothing system



ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

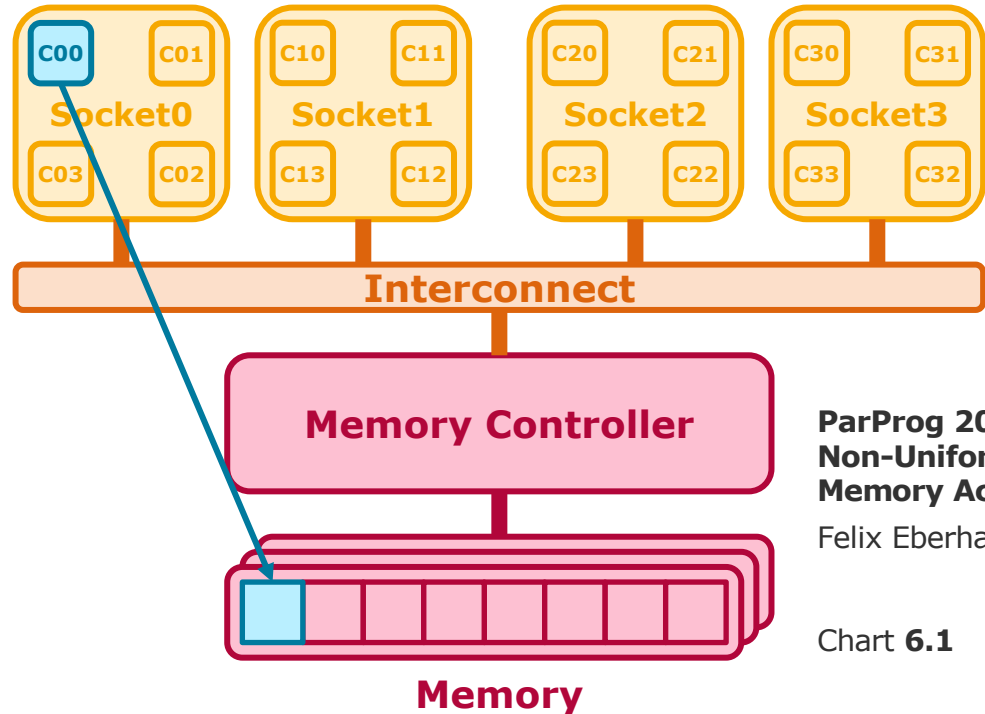
Chart 5

# Non-Uniform Memory Access

## Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

*Latency and bandwidth characteristic is equal for any pair of socket and memory location.*

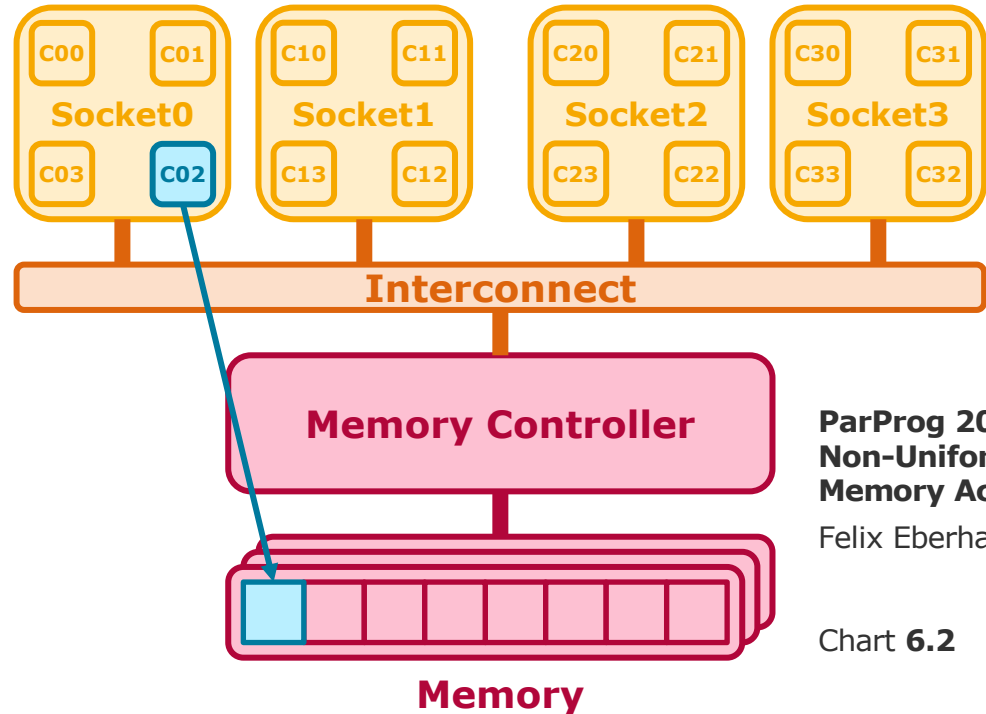


# Non-Uniform Memory Access

## Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

*Latency and bandwidth characteristic is equal for any pair of socket and memory location.*

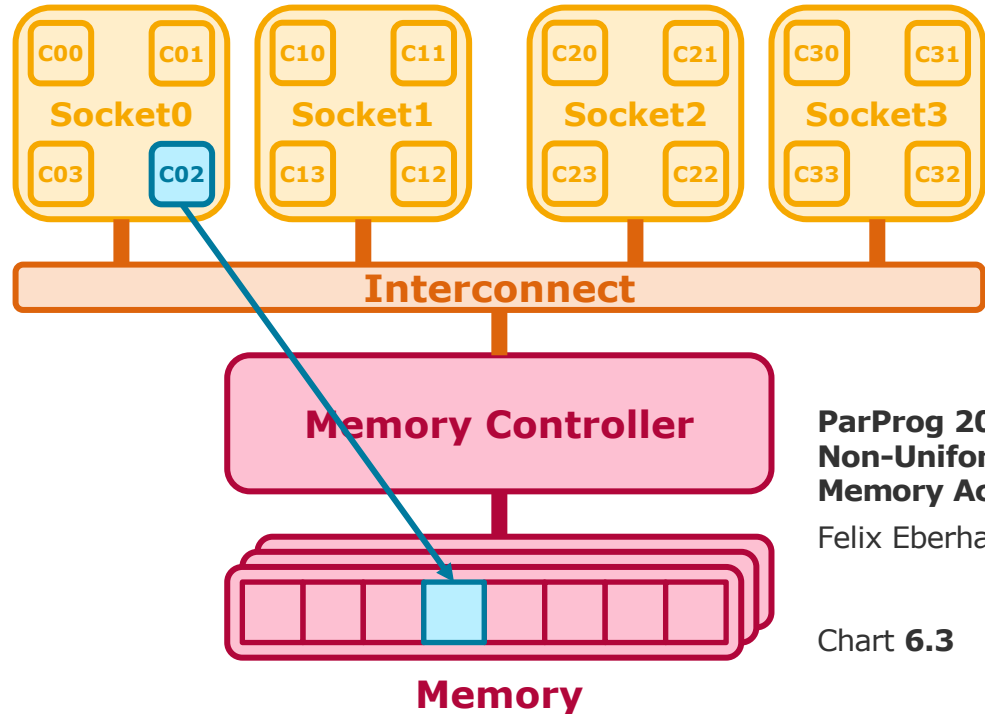


# Non-Uniform Memory Access

## Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

*Latency and bandwidth characteristic is equal for any pair of socket and memory location.*



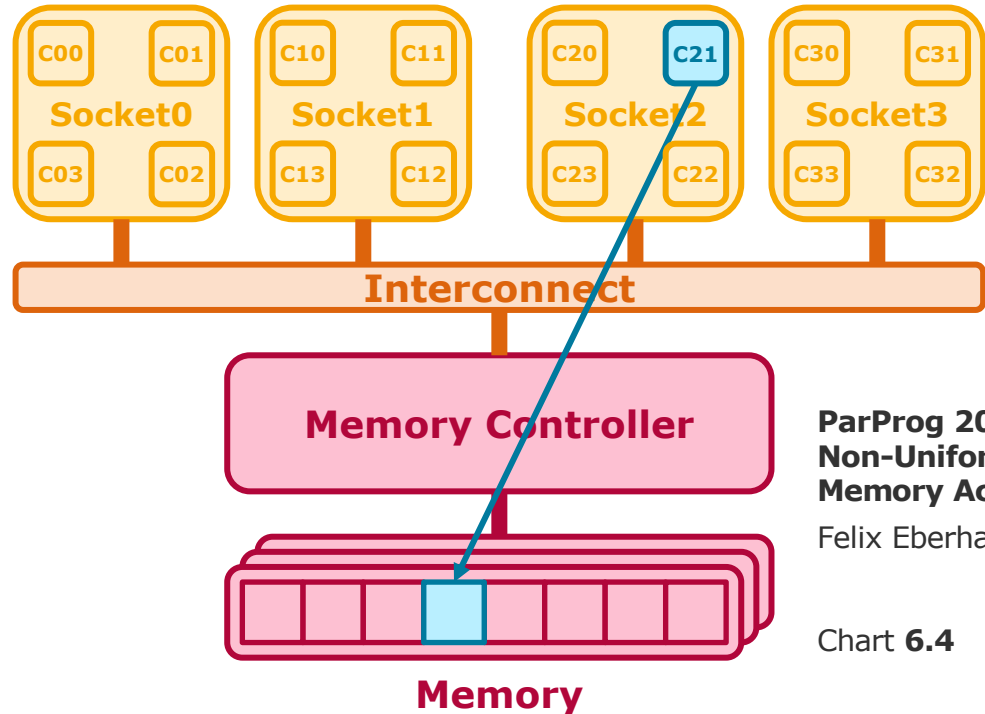


# Non-Uniform Memory Access

## Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

*Latency and bandwidth characteristic is equal for any pair of socket and memory location.*

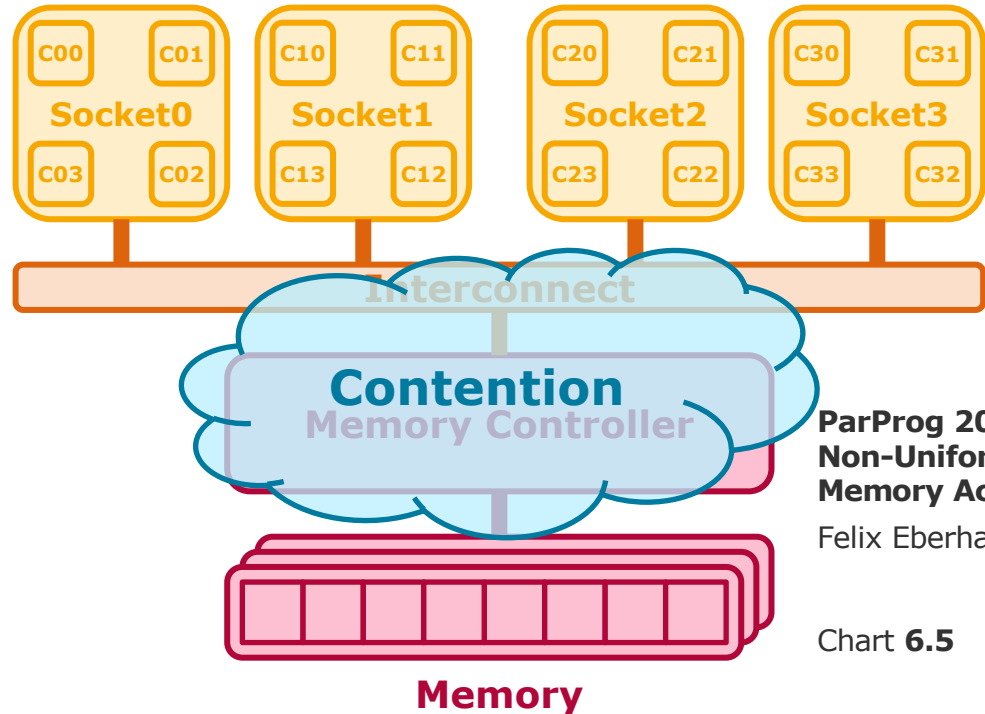


# Non-Uniform Memory Access

## Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

*Latency and bandwidth characteristic is equal for any pair of socket and memory location.*

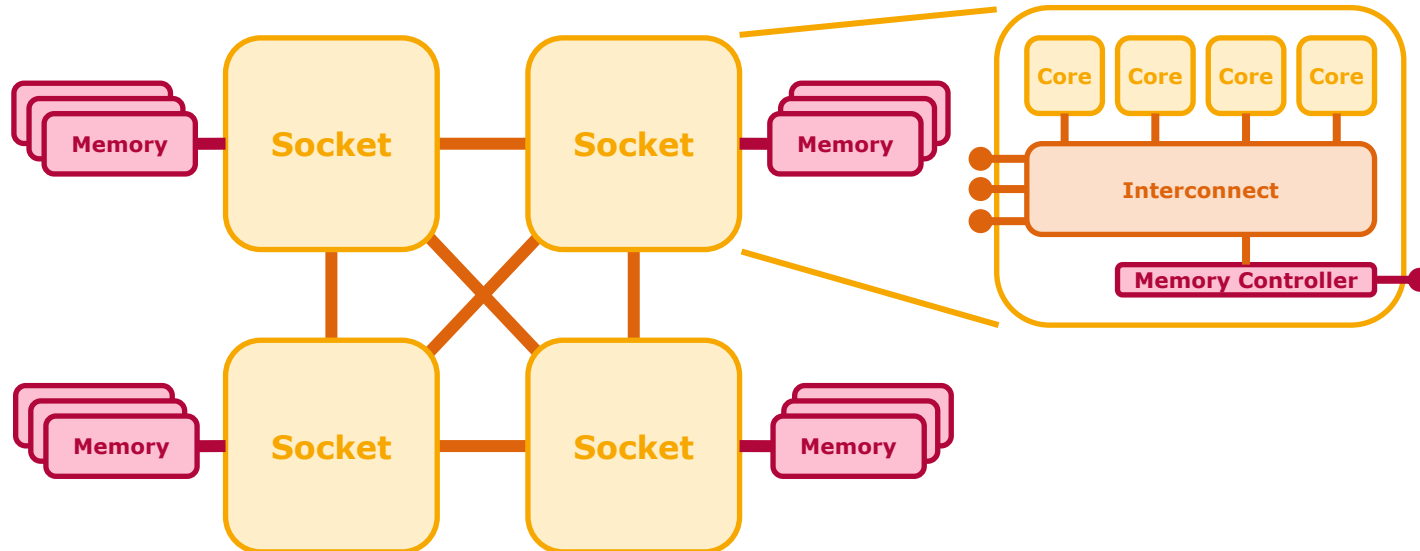


ParProg 2020 B4  
Non-Uniform  
Memory Access  
Felix Eberhardt

Chart 6.5

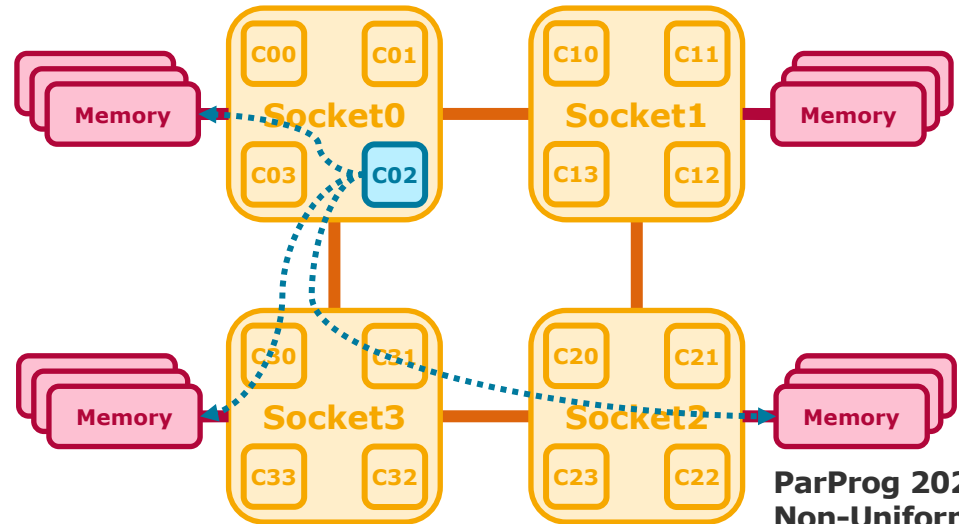
# Non-Uniform Memory Access Concept

- Part of the main memory is directly attached to a socket (**local memory**)
- Memory attached to a different socket can be accessed indirectly via the other socket's memory controller and interconnect (**remote memory**)
- Socket + local memory form a **NUMA node**



# Non-Uniform Memory Access Characteristics

- Local memory access does not involve inter-socket links, but they are shared for remote requests
  - Local performance can suffer from remote activity
- Remote memory access involves one or more inter-socket links, as they need not form a complete graph
  - Access to different remote memory regions is non-uniform as well



**ParProg 2020 B4  
Non-Uniform  
Memory Access**

Felix Eberhardt

# Non-Uniform Memory Access Concept

- Multiple **point to point links** between sockets **scale better** than a shared interconnect
- **Multiple memory controllers** partition address space and provide a **higher** total memory **bandwidth**  
(though the bandwidth to a single local region remains the same)
- Access to local memory behaves exactly like UMA system
- Access to remote memory traverses more hops (*local interconnect* → *inter-socket link* → *remote interconnect* → *remote memory controller*)
  - Certainly higher access latency
  - Probably lower bandwidth, as inter-socket link is likely not as wide as on chip connections
- **Predominant architecture for current multi-socket machines**

# Non-Uniform Memory Access Terminology

## Physical Perspective

1. Hardware Thread
2. Core
3. Chip, Die
4. Multichip Module
5. Socket, Package, Processor, CPU
6. Mainboard
7. Machine, System

## Logical Perspective

- Core, CPU, Processing Unit, Processing Element
- NUMA Node/Region

←

←

←

# Non-Uniform Memory Access

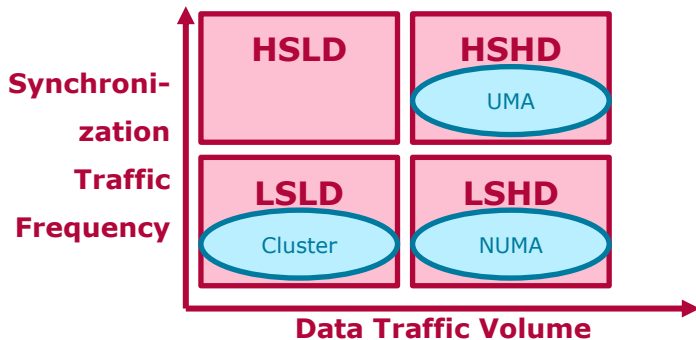
## Example: SGI UV 300H



- 240 Cores
- 12 TB RAM
- 16 Sockets

*What is a Killer Application for such a machine?*

- In-Memory Databases!



[Workload Taxonomy by Pfister]

ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

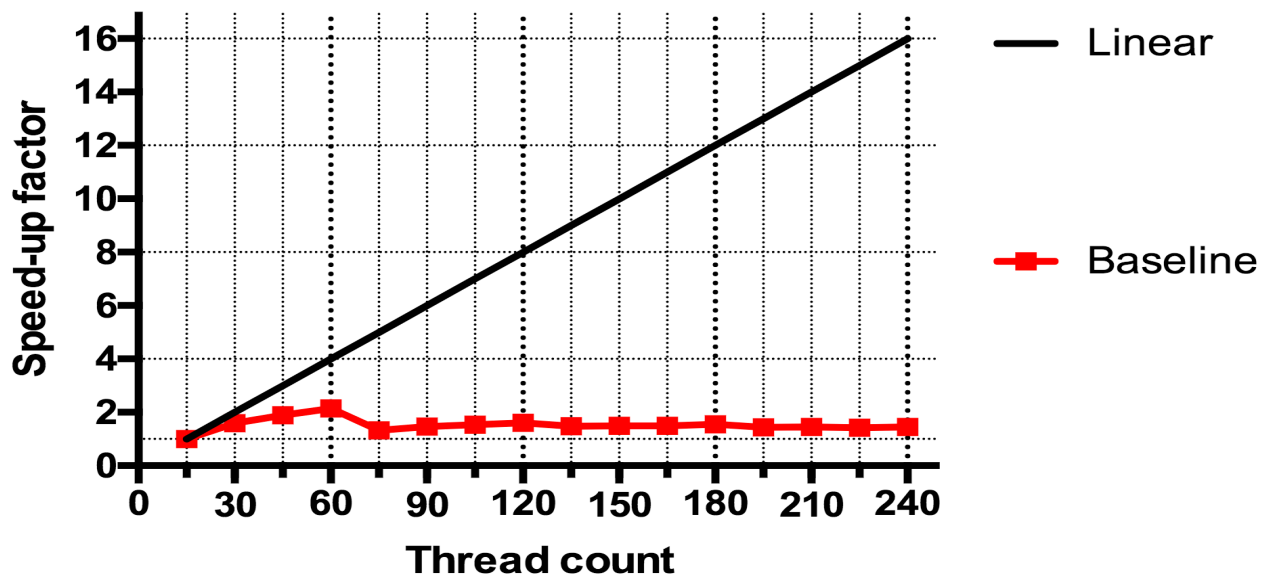
Chart 11

# Non-Uniform Memory Access

## Example: SGI UV 300H

### Experiment: NUMA behavior when scaling a workload

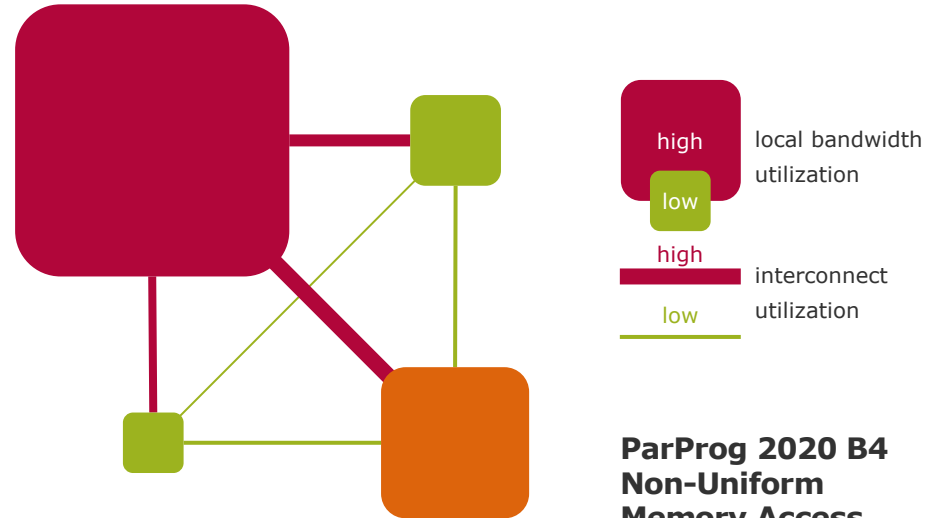
- Machine has 16 sockets x 15 cores x 2-way SMT (allocated in locality order)
- Performance **degrades** when using more than two sockets!





# Non-Uniform Memory Access Characteristics

- Unsuitable access patterns can severely degrade performance:
  - Inter-socket link contention on excessive remote memory accesses
  - Local memory controller contention on excessive combined local and remote memory accesses
  - Local interconnect contention also on excessive multi-hop forward traffic



**ParProg 2020 B4  
Non-Uniform  
Memory Access**

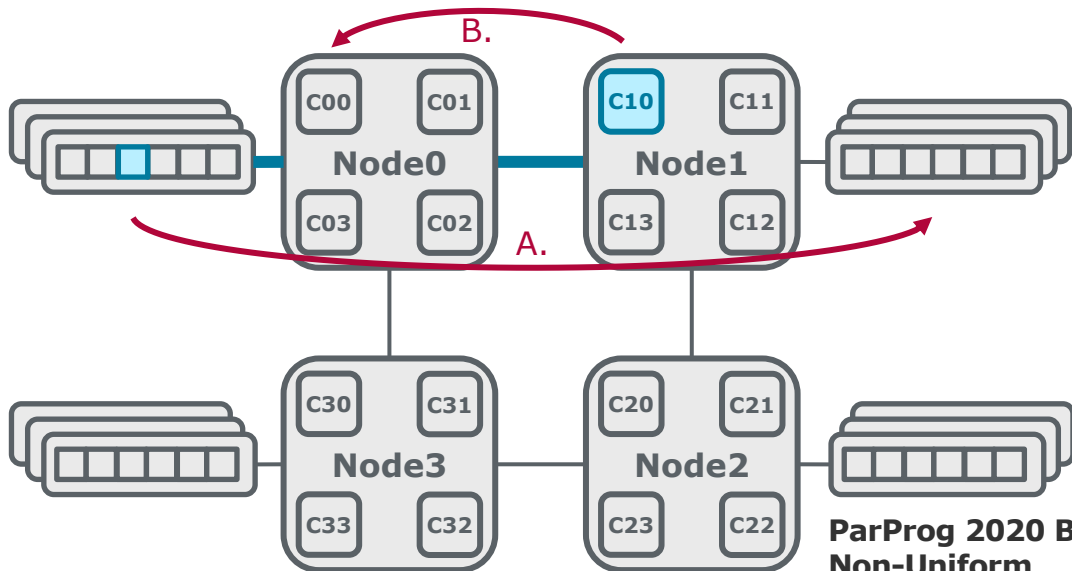
Felix Eberhardt

Chart 13

# Non-Uniform Memory Access Data Access Patterns

## Single task accesses private buffer on a different node

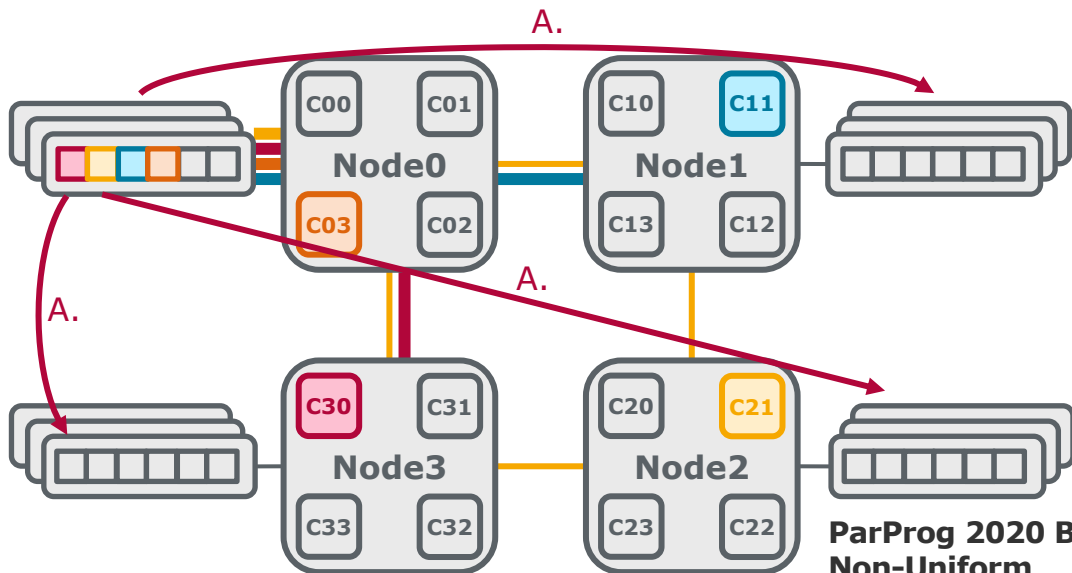
- A. Relocate remote buffer to local memory
- B. Relocate task to remote node
- Reduce inter-socket contention



# Non-Uniform Memory Access Data Access Patterns

## Multiple tasks on multiple nodes access private buffers on single node

- A. Relocate remote buffers to local memory
- Reduce memory controller contention

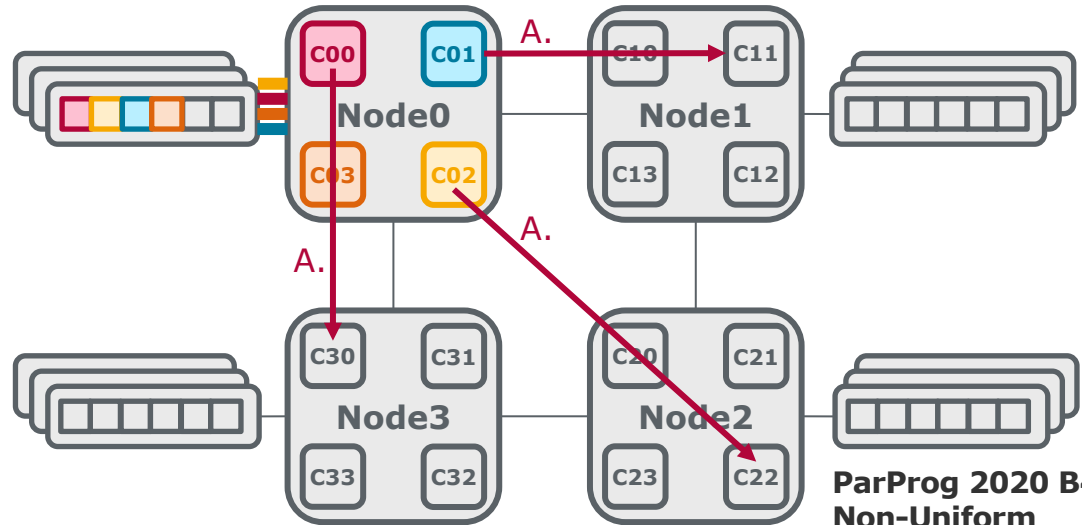


ParProg 2020 B4  
Non-Uniform  
Memory Access  
Felix Eberhardt

# Non-Uniform Memory Access Data Access Patterns

## Multiple tasks on a single node access private buffers on the same node

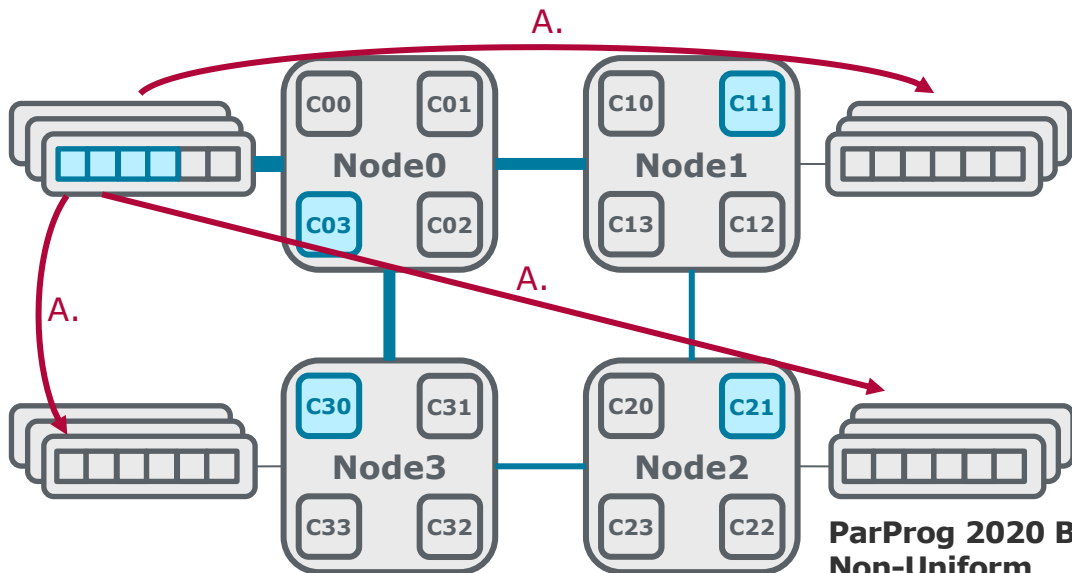
- A. Distribute tasks and buffers to different nodes
- Balance memory controller utilization



# Non-Uniform Memory Access Data Access Patterns

## Tasks on multiple nodes access a shared buffer on single node

- A. Distribute shared buffer among all nodes
- Reduce memory controller contention
- Balance inter-node traffic

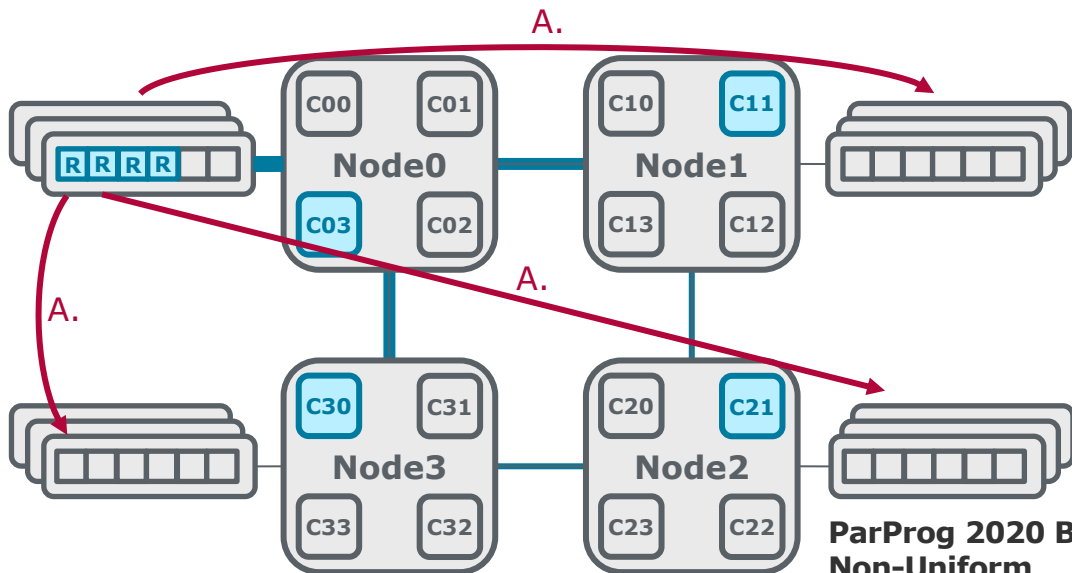


ParProg 2020 B4  
Non-Uniform  
Memory Access  
Felix Eberhardt

# Non-Uniform Memory Access Data Access Patterns

## Tasks on multiple nodes read a shared buffer on single node

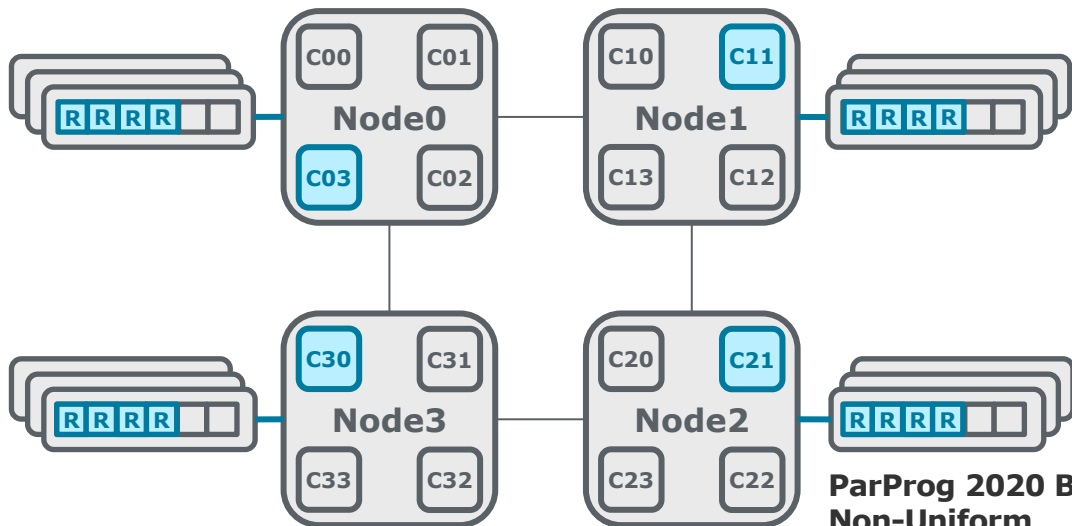
- A. Read only: Duplicate buffer on every node
- Avoid inter node traffic entirely



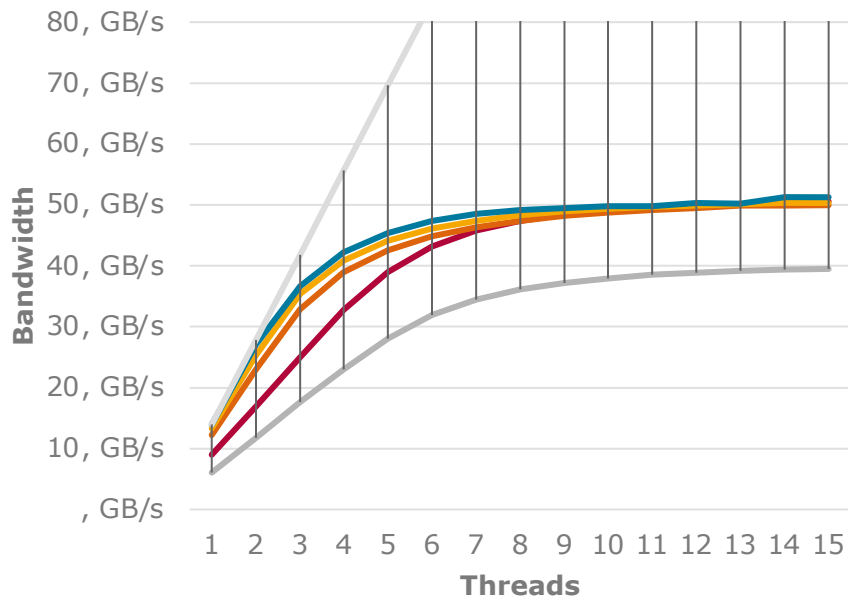
# Non-Uniform Memory Access Data Access Patterns

## Tasks on multiple nodes read a shared buffer on single node

- A. Read only: Duplicate buffer on every node
- Avoid inter node traffic entirely



# Non-Uniform Memory Access Local Bandwidth Characteristics



## Experiment on SGI UV 300H:

Threads on a single socket generate independent memory traffic

- Significant flattening of the curve after 6~8 active threads
- Local memory bandwidth exhausted, scaling beyond 8 threads has no benefits

ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 19





# Non-Uniform Memory Access System Bandwidth Characteristics

## Experiments on SGI UV 300 H:

memory on single node accessed by threads on local node	51.1 GB/s
memory on single node accessed by threads on local and one remote node	56.5 GB/s
memory on all 16 nodes accessed by threads on local nodes	816.0 GB/s
memory on all 16 nodes accessed by threads on local and remote nodes (random pattern)	185.0 GB/s

110.6%

1597.5% ~ ×16

22.7%

- **Huge performance potential, provided thread and memory placement is chosen adequately**

ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 20

# Non-Uniform Memory Access Placement Decisions

## Avoid data movement

- Remote memory accesses across long distances take time → high latency → wasted cycles
- High volume will cause contention → high latency for accessing threads → wasted cycles

## Avoid contention

- Balance utilization of resources (memory controllers, interconnect, ...)

## Analyze data access patterns

- Decompose loosely coupled tasks → increase flexibility of placement
- Agglomerate tightly coupled tasks → reduce communication overhead
- Identify shared and private data chunks and place accordingly
- Identify read-only, read-write, write-only access patterns
- Consider benefits of dynamic adaption during runtime

## ➤ Maximize data locality

**ParProg 2020 B4  
Non-Uniform  
Memory Access**

Felix Eberhardt

Chart 21

## Tradeoff:

**computational load balancing**  $\diamond$  **data locality**

- Possible on different granularities (Process • Thread • Task)
- Realized in the OS through an **Affinity Mask**:  
*A bitmask to specify on which logical cpu the process or threads in a process can be scheduled*
  - **Pinning** (= only a single bit set)
- Affinity mask can be adjusted at runtime:
  - **Computation follows data**

# Non-Uniform Memory Access

## Data placement

- Placement granularity is a page (4k, 64k, ... 64GB)
  - **Static** at allocation time:
    - Placement policies or specific requests govern page location for every allocation*
    - First-touch – defacto standard policy
    - Allocate on fixed node(s)
    - Interleaving
    - (Page replication on multiple nodes, consistency!)
  - **Dynamic** at runtime:
    - Pages can migrate between different nodes after allocation*
- ***Data follows computation***

# Non-Uniform Memory Access - Toolbox

## External Placement Control

`numactl` wraps application and enforces specific placement policies

- **Thread Placement** set default affinity mask for a given process

- `numactl --physcpubind=<cpus>`
- `numactl --cpunodebind=<nodes>`

`<cpus>` is a comma delimited list of cpu numbers or A-B ranges or all

- `taskset` is another tool to control the affinity mask, able to modify affinity masks of running processes

- **Data Placement**

- `numactl --interleave=<nodes>`
- `numactl --membind=<nodes>`

`<nodes>` is a comma delimited list of node numbers or A-B ranges or all

# Non-Uniform Memory Access - Toolbox

## Internal Placement Control

### Thread Placement

#### Systemcall

- `sched_setaffinity(pid_t pid, size_t cpusetsize, cpu_set_t *mask)`

#### Pthread

- `pthread_setaffinity_np(pthread_t thread, size_t cpusetsize, const cpu_set_t *cpuset)`

#### libnuma

- `numa_run_on_node(int node)`

### Data Placement

#### libnuma

- `void *numa_alloc_onnode(size_t size, int node)`
- `void *numa_alloc_interleaved(size_t size)`
- `int numa_move_pages(int pid, unsigned long count, void **pages, const int *nodes, int *status, int flags);`

libnuma

>man 3 numa

**ParProg 2020 B4**  
**Non-Uniform**  
**Memory Access**

Felix Eberhardt

Chart **25**

# Non-Uniform Memory Access - Toolbox

## Experiment: First-Touch Placement Policy

- Thread visits all NUMA nodes in the system
- Allocates memory on current node and touches the memory on next node
- To determine location of memory page we use:

```
move_pages(pid, count, **pages, *nodes, *status, flags);
```

```
int main(void) {  
...  
int n = numa_max_node();  
for (int i = 1; i <= n; i++){  
...  
while(  
    numa_node_of_cpu(sched_getcpu()) != i){  
    sleep(1);  
}  
...  
check_address(array[0]);  
}
```

```
void check_address(void* addr){  
    int status[1] = { -1 };  
    int ret = move_pages( 0, 1, &addr, NULL, status, 0);  
...  
}
```

# Non-Uniform Memory Access - Toolbox Experiment: First-Touch Placement Policy

```
felix@ubuntu1804ppc64el:~$ ssh felix@192.168.42.88 -- 107x44
felix@ubuntu1804ppc64el:~$ ./FirstTouch
There are 8 nodes on your system, we assume they are continuous!
Page size in bytes: 65536
Begin Experiment

[0] ALLOC
[0] INFO Page(0x7010ecc10000) is not placed yet
[1] TOUCH
[1] INFO Page(0x7010ecc10000) is at node 1

[1] ALLOC
[1] INFO Page(0x7010ecc00000) is not placed yet
[2] TOUCH
[2] INFO Page(0x7010ecc00000) is at node 2

[2] ALLOC
[2] INFO Page(0x7010ec990000) is not placed yet
[3] TOUCH
[3] INFO Page(0x7010ec990000) is at node 3

[3] ALLOC
[3] INFO Page(0x7010ec980000) is not placed yet
[4] TOUCH
[4] INFO Page(0x7010ec980000) is at node 4

[4] ALLOC
[4] INFO Page(0x7010ec970000) is not placed yet
[5] TOUCH
[5] INFO Page(0x7010ec970000) is at node 5

[5] ALLOC
[5] INFO Page(0x7010ec960000) is not placed yet
[6] TOUCH
[6] INFO Page(0x7010ec960000) is at node 6

[6] ALLOC
[6] INFO Page(0x7010ec950000) is not placed yet
[7] TOUCH
[7] INFO Page(0x7010ec950000) is at node 7

[7] ALLOC
[7] INFO Page(0x7010ec940000) is not placed yet
felix@ubuntu1804ppc64el:~$
```

ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 27



# Non-Uniform Memory Access - Toolbox

## Topology Discovery

### Tools for topology discovery:

- ACPI distance values
- Linux sysfs
- Libnuma: numactl
- Hwloc lstopo
- MLC (Memory Latency Checker)
- ...

### Tools for analyzing the runtime behaviour:

- Intel Performance Counter Monitor
- numatop
- ...

```
felixeberhardt — felix.eberhardt@dl1580-1: ~/Lehre/parProg/numatop — ssh felix...
NumaTOP v2.0, (C) 2015 Intel Corporation

Monitoring the process "mgen" (82578) (interval: 5.0s)

  NODE  RPI(K)  LPI(K)  RMA(K)  LMA(K)  RMA/LMA  CPI  CPU%
  ----  -
  0     139.4    0.0   20341.6    0.6   36066.7  85.73  2.8
  1         0.0    0.0     0.0     0.0     0.0    0.00  0.0
  2         0.0    0.0     0.0     0.0     0.0    0.00  0.0
  3         0.0    0.0     0.0     0.0     0.0    0.00  0.0

CPU% = per-node CPU utilization
Q; H; B; R; N: Node; L: Latency; C: Call-Chain; O: LLC OCCUPANCY
```

numatop: top focused on NUMA-related information

# Non-Uniform Memory Access - Toolbox

## Topology Discovery: Linux sysfs

### Information provided:

- NUMA nodes
- ACPI distance values of nodes and cores
- Mapping of cores to nodes
- Cache sizes, levels, associativity, cache line size
- Cache sharing of CPUs
- Restrictions:
  - Linux only

```
Macintosh HD -- ssh -- 88x23
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0> ls
cache  cpufreq  crash_notes  node0  thermal_throttle  topology
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0> cd topology/
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/topology> cat core_siblings_list
0-14,240-254
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/topology> cd ../node0/
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/node0> cat distance
10 16 19 16 50 50 50 50 50 50 50 50 50 50 50
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/node0> cat cpulist
0-14,240-254
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/node0> cd ..
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0> cd topology/
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/topology> cat thread_siblings_list
0,240
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/topology> cd ../cache/
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/cache> ls
index0  index1  index2  index3
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/cache> cd index0
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/cache/index0> ls
coherency_line_size  physical_line_partition  size
level                shared_cpu_list          type
number_of_sets      shared_cpu_map           ways_of_associativity
Felix.Eberhardt@side:/sys/devices/system/cpu/cpu0/cache/index0>
```

# Non-Uniform Memory Access - Toolbox

## Topology Discovery: libnuma

---

- `numa_max_node()` get the number of the highest node in the system
- `numa_num_configured_nodes()` get the total number of NUMA nodes in the system
- `numa_num_configured_cpus()` get the total number of cores in the system
- `numa_distance(int node1, int node2)` get the distance between two nodes as reported by ACPI
- `numa_node_to_cpus(int node, struct bitmask *mask)` get a bitmask of all cores associated with the given NUMA node
- `numa_node_of_cpu(int cpu)` get the node associated with the given core id

# Non-Uniform Memory Access - Toolbox

## Topology Discovery: numactl

Information provided:

- NUMA Nodes
- ACPI distance values of nodes and cores
- Mapping of cores to nodes
  
- Restrictions:
- Linux only
- Available as library to be used in applications to query system devices

```
Macintosh HD -- ssh -- 88x23
node 15 cpus: 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 465 466 467 46
8 469 470 471 472 473 474 475 476 477 478 479
node 15 size: 753648 MB
node 15 free: 622078 MB
node distances:
node  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
0:  10  16  19  16  50  50  50  50  50  50  50  50  50  50  50  50
1:  16  10  16  19  50  50  50  50  50  50  50  50  50  50  50  50
2:  19  16  10  16  50  50  50  50  50  50  50  50  50  50  50  50
3:  16  19  16  10  50  50  50  50  50  50  50  50  50  50  50  50
4:  50  50  50  50  10  16  19  16  50  50  50  50  50  50  50  50
5:  50  50  50  50  16  10  16  19  50  50  50  50  50  50  50  50
6:  50  50  50  50  19  16  10  16  50  50  50  50  50  50  50  50
7:  50  50  50  50  16  19  16  10  50  50  50  50  50  50  50  50
8:  50  50  50  50  50  50  50  50  10  16  19  16  50  50  50  50
9:  50  50  50  50  50  50  50  50  16  10  16  19  50  50  50  50
10: 50  50  50  50  50  50  50  50  19  16  10  16  50  50  50  50
11: 50  50  50  50  50  50  50  50  16  19  16  10  50  50  50  50
12: 50  50  50  50  50  50  50  50  50  50  50  50  10  16  19  16
13: 50  50  50  50  50  50  50  50  50  50  50  50  16  10  16  19
14: 50  50  50  50  50  50  50  50  50  50  50  50  19  16  10  16
15: 50  50  50  50  50  50  50  50  50  50  50  50  16  19  16  10
Felix.Eberhardt@side:~>
```

**ParProg 2020 B4**  
**Non-Uniform**  
**Memory Access**

Felix Eberhardt

Chart **31**

# Non-Uniform Memory Access - Toolbox

## Topology Discovery: hwloc / Istopo

Information provided:

- NUMA Nodes
- ACPI distance values of nodes and cores
- Mapping of cores to nodes
- Grouping of nodes according to distance values
- Memory hierarchy (Caches)

Restrictions:

- Several platforms: Windows, Linux, BSD, ...
- Available as library to be used in applications to query system devices



ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 32

# Non-Uniform Memory Access - Toolbox

## Topology Discovery: Memory Latency Checker

Empirical information provided:

- Latencies to local memory hierarchy
- Bandwidth to local memory hierarchy
- Latencies between NUMA nodes
- Bandwidth between NUMA nodes
- Latencies of Cache-to-Cache transfers
- Latencies under load

Restrictions:

- Only on Intel Processors
- No source code available

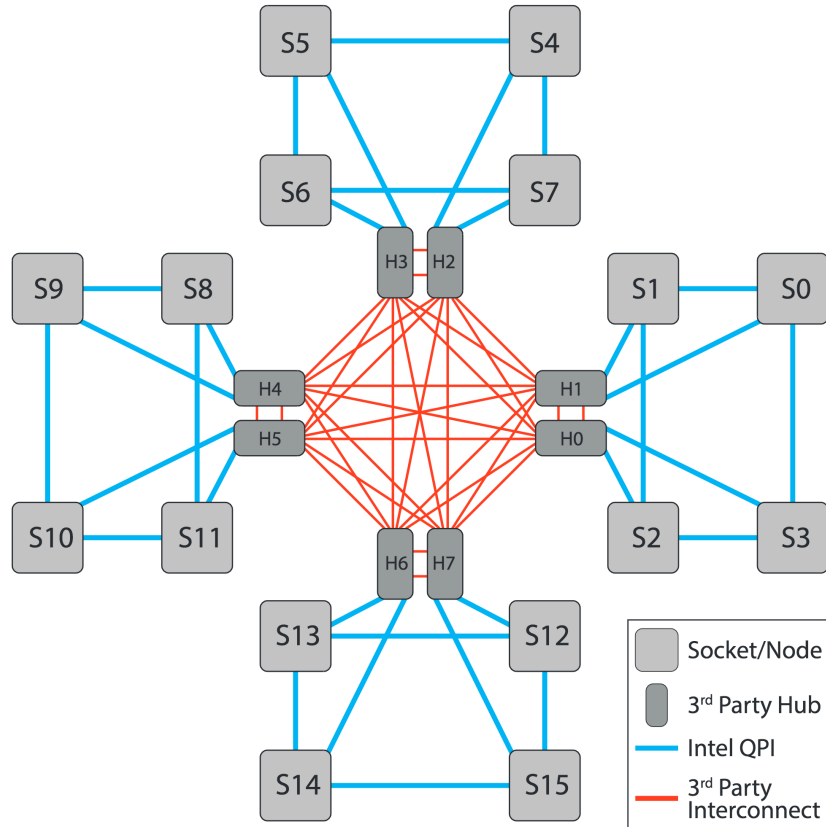
```
felixeberhardt — felix.eberhardt@dl580-1: ~/Lehre/parPro...
[$sudo ./mlc --latency_matrix
Intel(R) Memory Latency Checker - v3.6
Command line parameters: --latency_matrix

Using buffer size of 200.000MiB
Measuring idle latencies (in ns)...
          Numa node
Numa node  0      1      2      3
          0    125.9  252.8  252.9  263.3
          1    247.5  123.5  246.9  247.5
          2    248.1  246.9  123.3  247.6
          3    249.1  247.0  246.9  123.4
$
```

**ParProg 2020 B4**  
**Non-Uniform**  
**Memory Access**

Felix Eberhardt

# Non-Uniform Memory Access Topology Examples: SGI UV-300H



ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 34

# Non-Uniform Memory Access Topology Examples: SGI UV300H

## ACPI Distance Values

- Can be acquired with `numactl --hardware`
- Clusters relate to blades in the system
- Seem to be related to latency and bandwidth characteristics (see next slide)

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	10	16	19	16	50	50	50	50	50	50	50	50	50	50	50	50
2	16	10	16	19	50	50	50	50	50	50	50	50	50	50	50	50
3	19	16	10	16	50	50	50	50	50	50	50	50	50	50	50	50
4	16	19	16	10	50	50	50	50	50	50	50	50	50	50	50	50
5	50	50	50	50	10	16	19	16	50	50	50	50	50	50	50	50
6	50	50	50	50	16	10	16	19	50	50	50	50	50	50	50	50
7	50	50	50	50	19	16	10	16	50	50	50	50	50	50	50	50
8	50	50	50	50	16	19	16	10	50	50	50	50	50	50	50	50
9	50	50	50	50	50	50	50	50	10	16	19	16	50	50	50	50
10	50	50	50	50	50	50	50	50	16	10	16	19	50	50	50	50
11	50	50	50	50	50	50	50	50	19	16	10	16	50	50	50	50
12	50	50	50	50	50	50	50	50	16	19	16	10	50	50	50	50
13	50	50	50	50	50	50	50	50	50	50	50	50	10	16	19	16
14	50	50	50	50	50	50	50	50	50	50	50	50	16	10	16	19
15	50	50	50	50	50	50	50	50	50	50	50	50	19	16	10	16
16	50	50	50	50	50	50	50	50	50	50	50	50	16	19	16	10

ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 35



# Non-Uniform Memory Access Topology Examples: SGI UV-300H

## Measured Latency

- Intel MLC used
- Clusters relate to blades in the system
- 3 classes of latencies:
  - Local: ~110 ns
  - Neighbor: ~200 ns
  - Blade: ~230 ns
  - Far remote: ~480 ns

Factor of ~4x between local and far remote!

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	111	197	231	204	482	484	481	485	485	489	487	489	488	489	488	488
2	203	111	191	234	481	481	480	482	482	486	486	488	486	489	487	488
3	242	197	112	197	482	484	480	484	486	488	485	488	488	488	487	488
4	212	233	189	113	481	481	480	482	484	488	483	488	486	488	487	488
5	481	482	480	483	114	194	225	200	480	481	481	487	483	489	487	488
6	481	482	480	483	197	112	192	232	480	481	481	487	483	489	487	488
7	482	484	481	484	234	195	111	192	480	483	481	488	487	489	486	487
8	481	483	480	482	203	230	190	113	480	481	480	488	485	489	485	486
9	487	488	484	488	481	481	480	484	114	195	226	200	481	489	482	483
10	485	487	483	488	480	481	480	482	191	114	190	231	481	487	481	482
11	487	488	481	487	482	482	480	482	227	195	114	195	481	488	481	482
12	488	488	483	488	482	483	480	483	198	229	189	114	481	488	482	483
13	488	490	486	489	485	488	487	488	481	485	483	488	112	193	227	200
14	487	488	484	488	484	488	485	489	480	483	482	488	190	114	191	230
15	487	489	485	489	485	489	483	489	481	485	482	488	227	194	112	197
16	487	488	484	488	485	488	483	488	480	485	481	487	195	227	190	113

ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 36

# Non-Uniform Memory Access Topology Examples: SGI UV-300H

## Measured Bandwidth

- Intel MLC used
- Clusters relate to blades in the system
- 3 classes of distances:
  - Local: ~51 GB/s
  - Neighbour: ~12.5 GB/s
  - Blade: ~11.5 GB/s
  - Far remote: 11.3 GB/s

Difference between remote nodes and far remote nodes not that big. However local and remote have a factor of ~4x in between!

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	51290	12757	11689	12102	11387	11393	11266	11353	11270	11394	11387	11246	11105	11224	11271	11252
2	12742	51674	12107	11496	11383	11370	11274	11375	11280	11339	11373	11225	11125	11228	11279	11260
3	12082	12127	51769	12739	11406	11356	11270	11361	11273	11396	11367	11252	11109	11239	11275	11257
4	12081	11512	12731	51349	11380	11364	11273	11376	11277	11347	11368	11249	11110	11244	11276	11260
5	11146	11372	11221	11370	51266	12756	11706	12095	11285	11389	11381	11235	11112	11214	11278	11252
6	11143	11369	11253	11367	12732	51729	12135	11474	11282	11351	11379	11229	11136	11212	11274	11254
7	11141	11374	11251	11357	12088	12120	51658	12732	11277	11402	11373	11227	11109	11201	11270	11250
8	11144	11367	11249	11363	12107	11511	12757	51628	11283	11365	11379	11239	11109	11233	11276	11256
9	11139	11399	11219	11358	11372	11360	11271	11400	51255	12733	11710	12117	11121	11223	11279	11261
10	11141	11375	11230	11360	11373	11364	11258	11364	12760	51257	12126	11481	11121	11243	11284	11266
11	11142	11367	11223	11367	11371	11358	11270	11373	12139	12134	51327	12741	11129	11246	11291	11264
12	11139	11394	11238	11383	11384	11387	11270	11378	12143	11512	12761	51261	11141	11221	11280	11259
13	11133	11374	11244	11355	11403	11389	11254	11405	11268	11378	11361	11219	51797	12737	11711	12097
14	11139	11362	11219	11377	11392	11402	11255	11373	11276	11387	11377	11238	12759	51246	12132	11513
15	11138	11398	11223	11381	11385	11389	11251	11369	11275	11346	11374	11251	12117	12127	51363	12762
16	11139	11391	11220	11373	11384	11397	11248	11380	11276	11305	11359	11252	12093	11512	12760	51383

ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 37

# Non-Uniform Memory Access Topology Examples: NUMA on Chip (Single Socket)

## ADVANCED PACKAGING

- 58mm x 75mm organic package
- 4 die-to-die Infinity Fabric links/die
  - 3 connected/die
- 2 SERDES/die to pins
  - 1/die to “top” (G) and “bottom” (P) links
  - Balanced I/O for 1P, 2P systems
- 2 DRAM-channels/die to pins

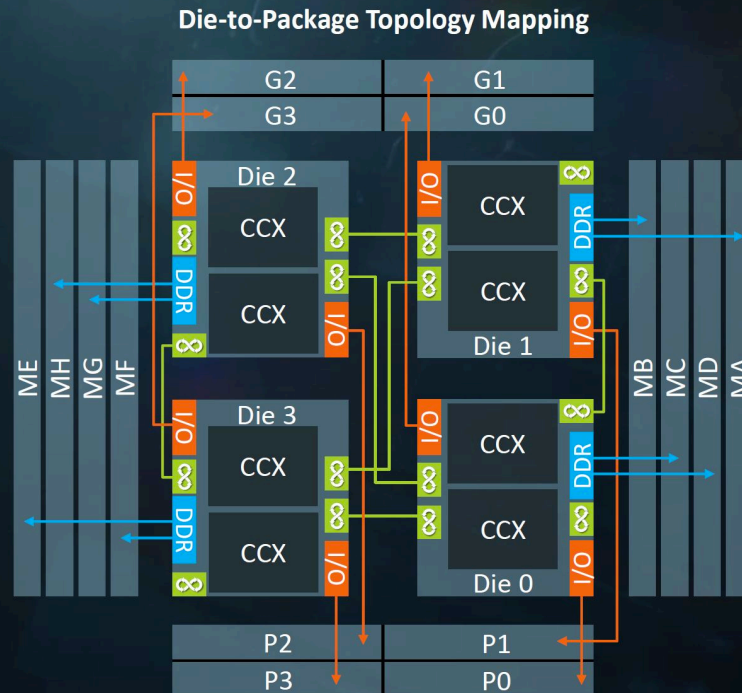
Purpose-built MCM Architecture

G[0-3], P[0-3] : 16 lane high-speed SERDES Links

M[A-H] : 8 x72b DRAM channels

∞ : Die-to-Die Infinity Fabric links

CCX: 4Core + L2/core + shared L3 complex



ParProg 2020 B4  
Non-Uniform  
Memory Access

Felix Eberhardt

Chart 38

# Non-Uniform Memory Access - Toolbox

## System Performance: numatop

Information provided:

- Similar to top tool
- Shows NUMA specific metrics
- Uses instruction sampling
- Memory view to find out which memory addresses are accessed frequently by remote nodes
- Ability to collect stack traces

Restrictions:

- Linux only, Kernel 3.9 or later

```
felixeberhardt — felix.eberhardt@dl580-1: ~/Lehre/parProg/numatop — ssh felix...
NumaTOP v2.0, (C) 2015 Intel Corporation

Monitoring memory areas (pid: 82578, AVG.LAT: 314ns, interval: 5.0s)

  ADDR      SIZE  ACCESS%  LAT(ns)  DESC
  -----
  7FB863F4F000 256M  100.0    314
  400000      136K   0.0      0    ../Lehre/parProg/numatop/mgen
  622000       4K   0.0      0    ../Lehre/parProg/numatop/mgen
  ..../parProg/numatop/mgen

felixeberhardt — felix.eberhardt@dl580-1: ~/Lehre/parProg/numatop — ssh felix...
NumaTOP v2.0, (C) 2015 Intel Corporation

Monitoring 1500 processes and 1646 threads (interval: 5.0s)

  PID      PROC      RMA(K)  LMA(K)  RMA/LMA  CPI  *CPU%
  -----
  82578    mgen      20332.7  0.8     26683.3  85.56  0.7
  82569    numatop   55.3    13.9    4.0      0.74   0.0
  8      ksoftirqd/0 13.6    7.1     1.9      2.95   0.0
  4062    zabbix_agen 23.9    7.9     3.0      1.99   0.0
  9      rcu_preempt 0.7     0.4     1.9      3.81   0.0
  5089    snmpd     5.8     4.0     1.5      2.66   0.0
  537    ksoftirqd/8 4.2     1.4     3.0      0.75   0.0
  30     ksoftirqd/3 1.3     0.0     212.8    52.06   0.0
  1      systemd  0.0     0.0     0.0      0.00   0.0
  2      kthreadd  0.0     0.0     0.0      0.00   0.0
  4      kworker/0:0 0.0     0.0     0.0      0.00   0.0
  7      mm_percpu_w 0.0     0.0     0.0      0.00   0.0
  10     rcu_sched  0.0     0.0     0.0      0.00   0.0
  11     rcu_bh     0.0     0.0     0.0      0.00   0.0
  12     migration/0 0.0     0.0     0.0      0.00   0.0
  13     watchdog/0 0.0     0.0     0.0      0.00   0.0
  14     cpuhp/0   0.0     0.0     0.0      0.00   0.0
  15     cpuhp/1   0.0     0.0     0.0      0.00   0.0
  16     watchdog/1 0.0     0.0     0.0      0.00   0.0
  17     migration/1 0.0     0.0     0.0      0.00   0.0
  18     ksoftirqd/1 0.0     0.0     0.0      0.00   0.0

[heap]
../libdl-2.23.so
../libdl-2.23.so
../libdl-2.23.so
../libdl-2.23.so
../libc-2.23.so
../libc-2.23.so
../libc-2.23.so
../libnuma.so.1.0.0
../libnuma.so.1.0.0
../libnuma.so.1.0.0
../libnuma.so.1.0.0
libpthread-2.23.so
libpthread-2.23.so
libpthread-2.23.so
libpthread-2.23.so
tribution

<- Hotkey for sorting: 1(RMA), 2(LMA), 3(RMA/LMA), 4(CPI), 5(CPU%) ->
CPU% = system CPU utilization
Q: Quit; H: Home; R: Refresh; I: IR Normalize; N: Node
```

ParProg 2020 B4  
Non-Uniform  
Memory Access  
Felix Eberhardt

# Non-Uniform Memory Access - Toolbox

## System Performance: Intel Processor Counter Monitor

### Information provided:

- API for Intel specific performance counters
- Core and Uncore events
- QPI links and memory controller utilization
- Many other tools available
  - PCIe
  - Cache allocation
  - ...
- <https://github.com/opcm/pcm>

### Restrictions:

- Available on Windows and Linux
- Intel processors only

```
Macintosh HD -- ssh -- 88x15
Time elapsed: 2 ms
Called sleep function for -1000 ms
-- NODE0 Memory (MB/s): 122.78 --||-- NODE1 Memory (MB/s): 96.83 --
-- NODE2 Memory (MB/s): 108.48 --||-- NODE3 Memory (MB/s): 34.88 --
-- NODE4 Memory (MB/s): 27.90 --||-- NODE5 Memory (MB/s): 30.37 --
-- NODE6 Memory (MB/s): 15.58 --||-- NODE7 Memory (MB/s): 22.02 --
-- NODE8 Memory (MB/s): 15.62 --||-- NODE9 Memory (MB/s): 17.57 --
-- NODE10 Memory (MB/s): 16.99 --||-- NODE11 Memory (MB/s): 19.49 --
-- NODE12 Memory (MB/s): 18.72 --||-- NODE13 Memory (MB/s): 20.38 --
-- NODE14 Memory (MB/s): 30.88 --||-- NODE15 Memory (MB/s): 19845.82 --
-----||-----
-- System Read Throughput(MB/s): 10757.21 --
-- System Write Throughput(MB/s): 9687.10 --
-- System Memory Throughput(MB/s): 20444.32 --
-----
```

```
Macintosh HD -- ssh -- 88x26
Intel(r) QPI traffic estimation in bytes (data and non-data traffic outgoing from CPU/socket through QPI links):
```

	QPI0	QPI1	QPI2		QPI0	QPI1	QPI2
SKT 0	114 M	29 M	30 M		0%	0%	0%
SKT 1	46 M	53 M	24 M		0%	0%	0%
SKT 2	47 M	36 M	27 M		0%	0%	0%
SKT 3	33 M	35 M	44 M		0%	0%	0%
SKT 4	61 M	15 M	22 M		0%	0%	0%
SKT 5	62 M	22 M	15 M		0%	0%	0%
SKT 6	46 M	16 M	19 M		0%	0%	0%
SKT 7	19 M	16 M	48 M		0%	0%	0%
SKT 8	89 M	17 M	22 M		0%	0%	0%
SKT 9	47 M	37 M	21 M		0%	0%	0%
SKT 10	35 M	29 M	22 M		0%	0%	0%
SKT 11	28 M	26 M	35 M		0%	0%	0%
SKT 12	44 M	17 M	22 M		0%	0%	0%
SKT 13	34 M	32 M	31 M		0%	0%	0%
SKT 14	32 M	25 M	31 M		0%	0%	0%
SKT 15	25 M	25 M	48 M		0%	0%	0%

```
-----
Total QPI outgoing data and non-data traffic: 1679 M
```