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

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

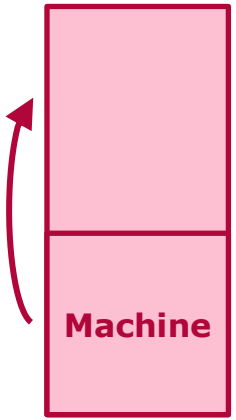
Recap
Optimization Goals

- 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

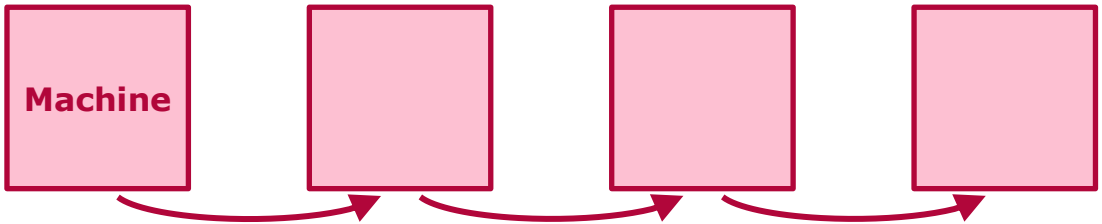
Non-Uniform Memory Access
Context: Scalability

- 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



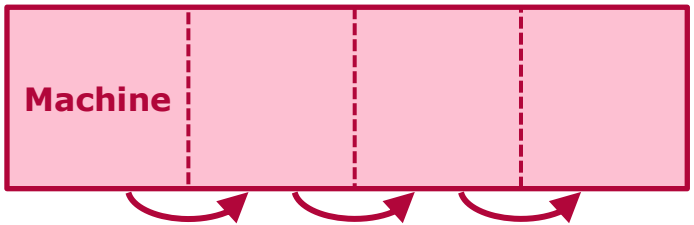
Non-Uniform Memory Access
Context: Scalability

- 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

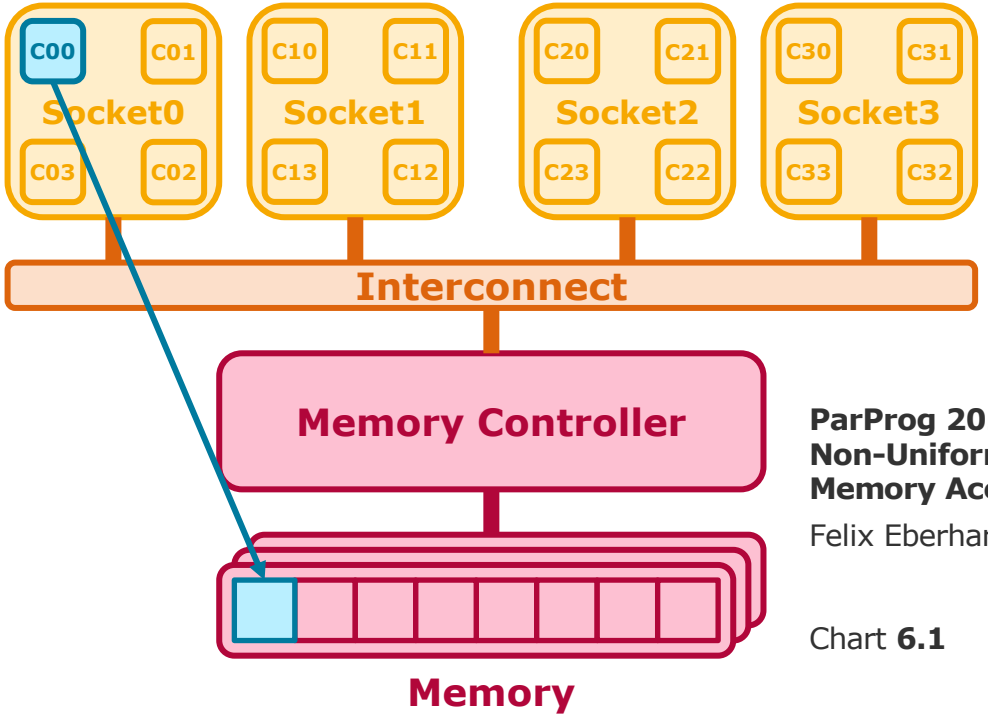
- Recent coherent interconnect technologies enable **hybrid** systems with both scale-up and scale-out characteristics:
  - Example: Gen-Z strives to connect an entire datacenter of machines coherently
  - ➢ User perceives a shared-memory system, but with the performance characteristics (communication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



Non-Uniform Memory Access
Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

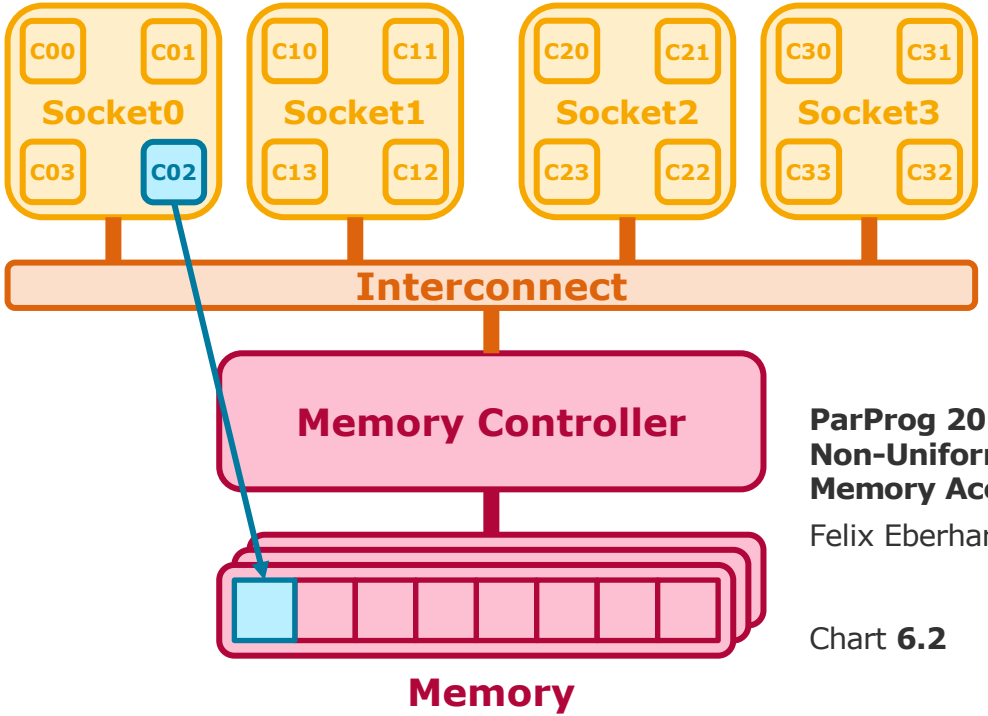
Latency and bandwidth characteristic is equal for any pair of socket and memory location.



Non-Uniform Memory Access
Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

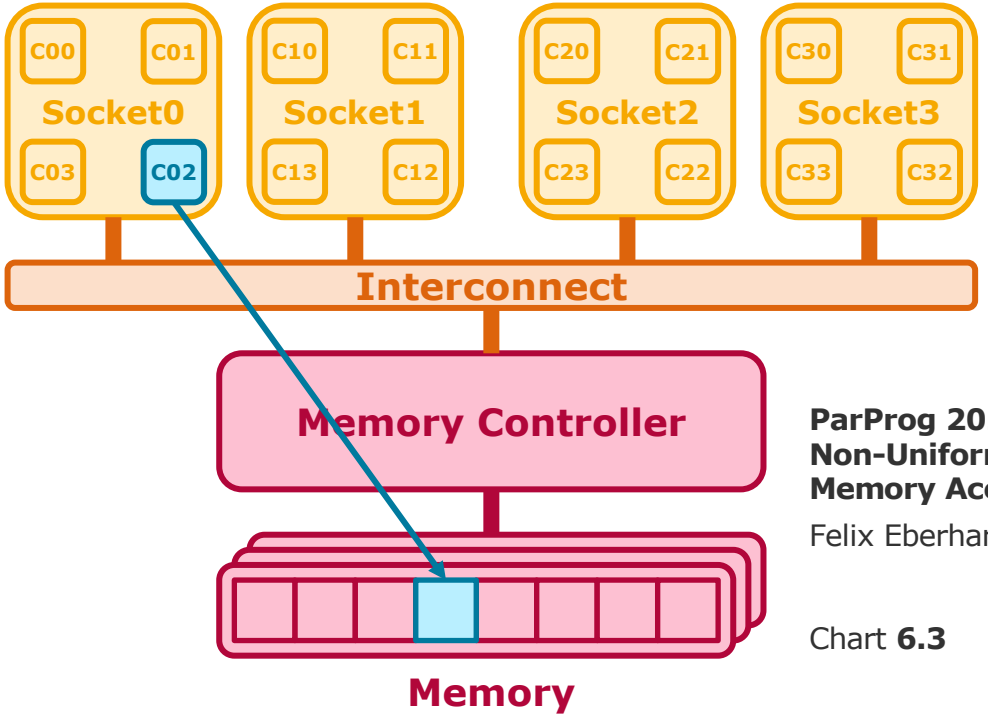
Latency and bandwidth characteristic is equal for any pair of socket and memory location.



Non-Uniform Memory Access
Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

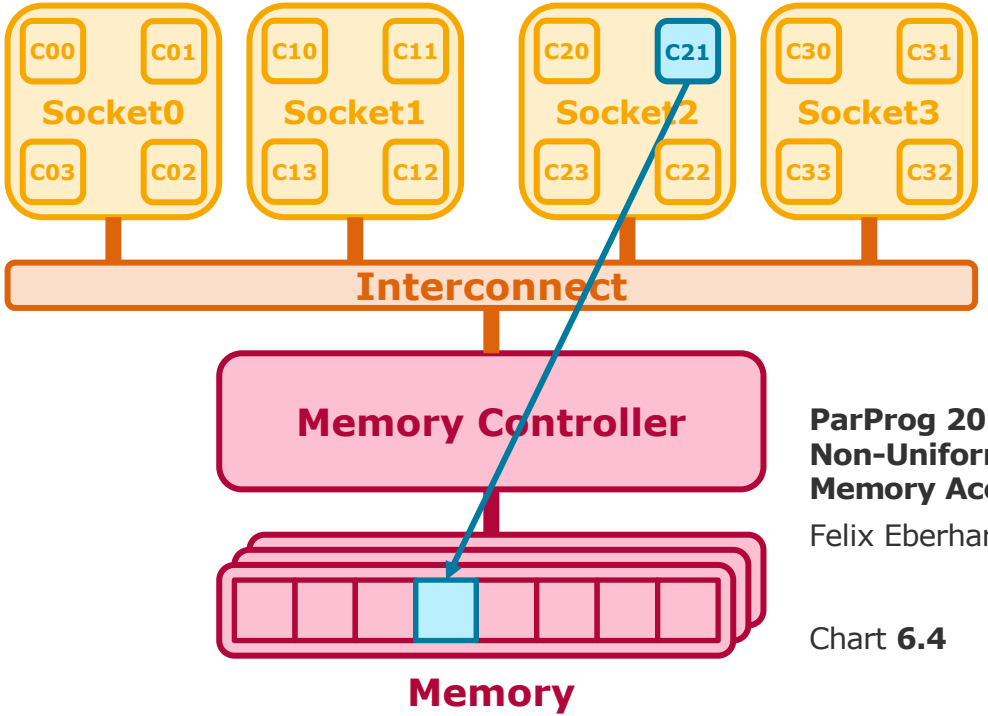
Latency and bandwidth characteristic is equal for any pair of socket and memory location.



Non-Uniform Memory Access
Context: Uniform Memory Access Machines

Multiple sockets access main memory through a shared interconnect.

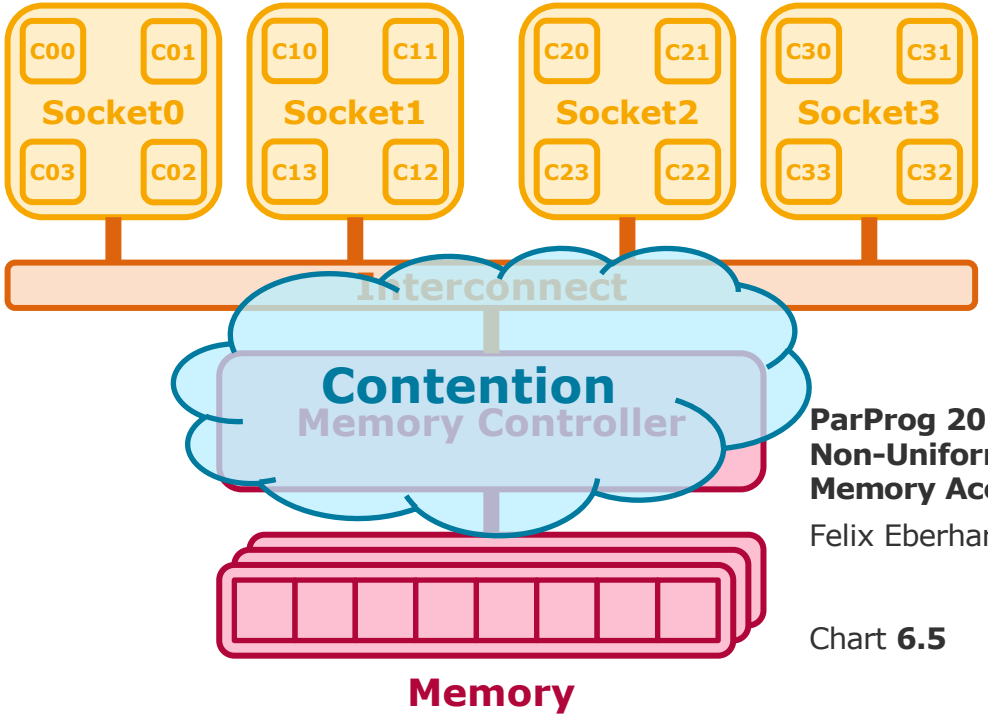
Latency and bandwidth characteristic is equal for any pair of socket and memory location.



Non-Uniform Memory Access
Context: Uniform Memory Access Machines

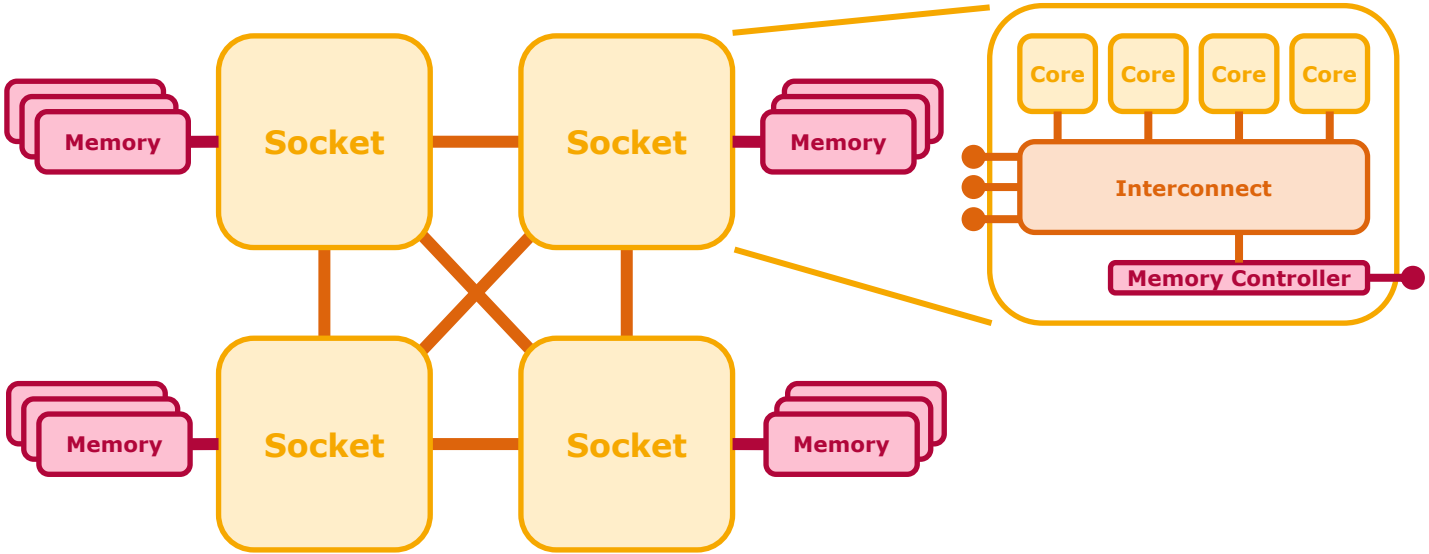
Multiple sockets access main memory through a shared interconnect.

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



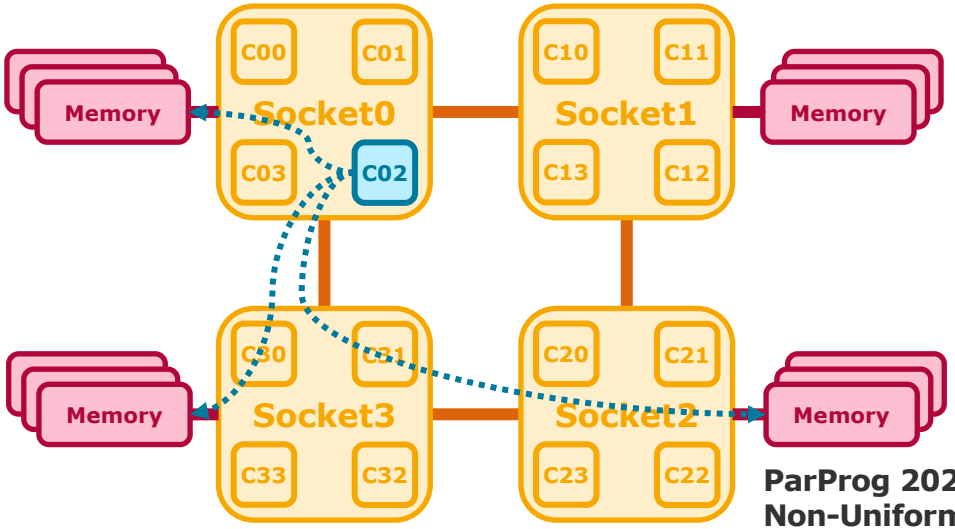
Non-Uniform Memory Access Concept

- 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



Non-Uniform Memory Access Concept

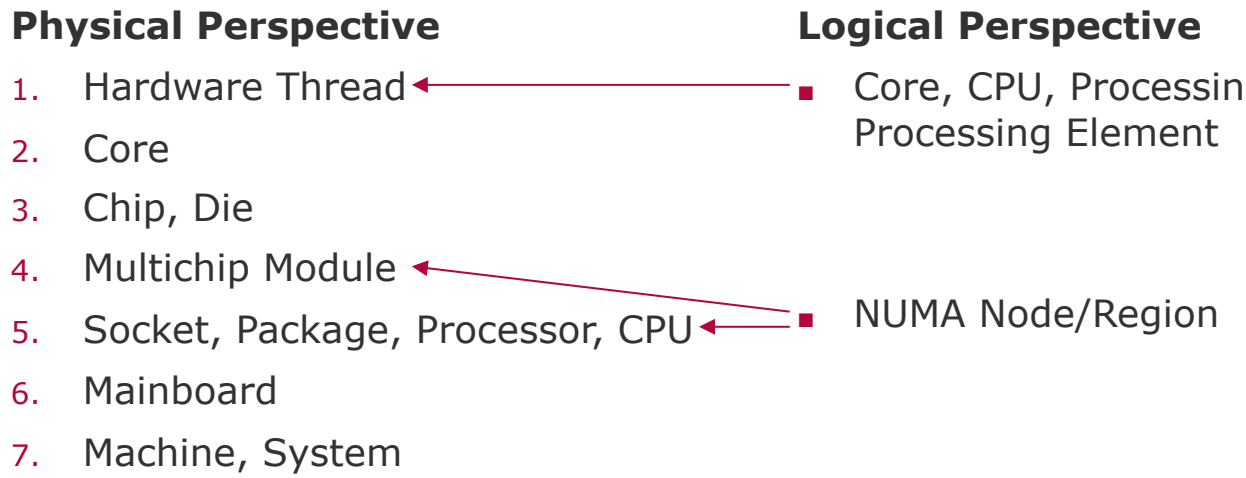
- Multiple **point to point links** between sockets **scale better** than a shared interconnect
- **Multiple memory controllers** partition address space and provide a **higher** total memory **bandwidth**

  (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

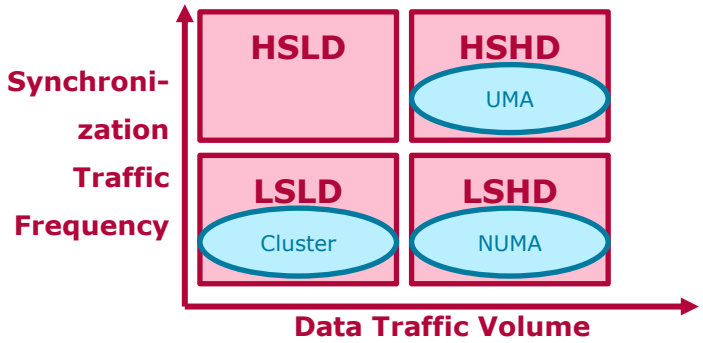


Non-Uniform Memory Access
Example: SGI UV 300H

- 240 Cores
- 12 TB RAM
- 16 Sockets

What is a Killer Application for such a machine?

- In-Memory Databases!

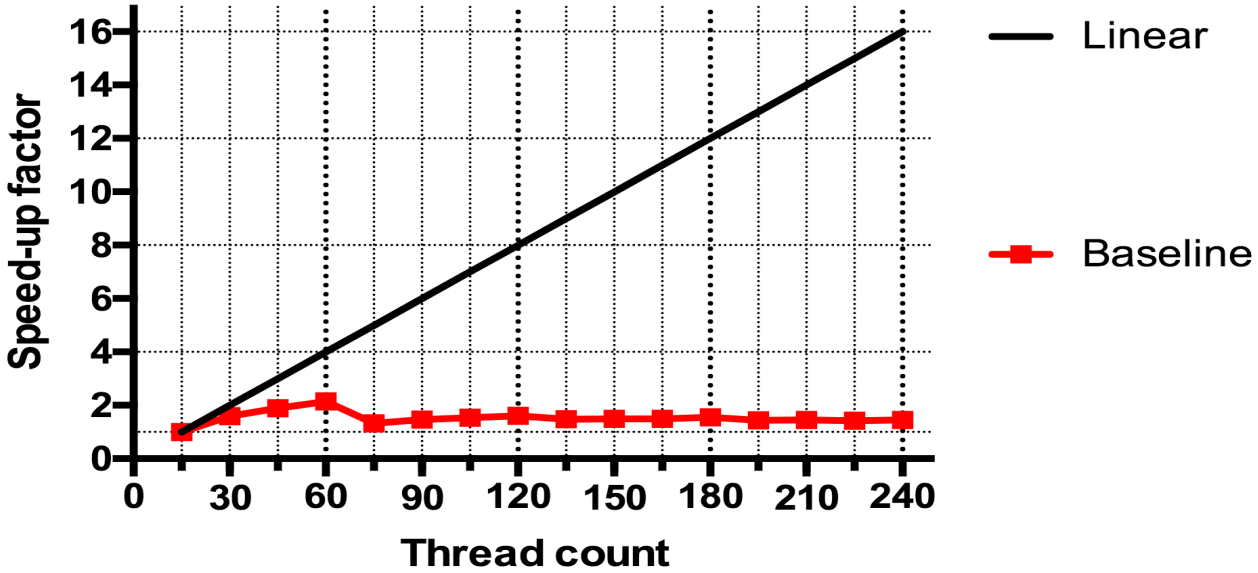


[Workload Taxonomy by Pfister]

Non-Uniform Memory Access
Example: SGI UV 300H

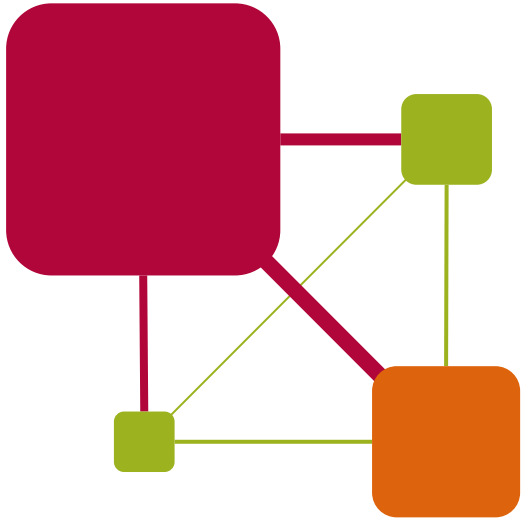
Experiment: NUMA behavior when scaling a workload

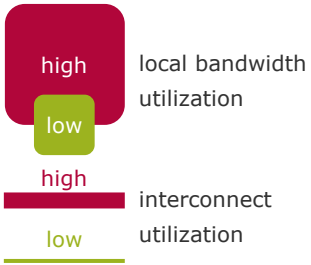
- Machine has 16 sockets x 15 cores x 2-way SMT (allocated in locality order)
- Performance **degrades** when using more than two sockets!



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





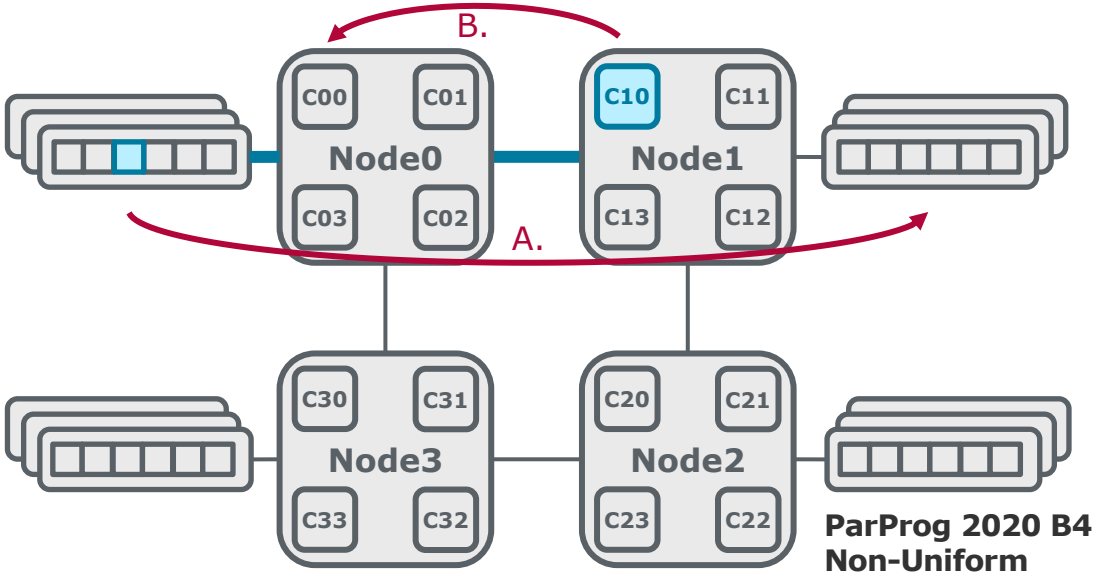
Non-Uniform Memory Access
Data Access Patterns

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

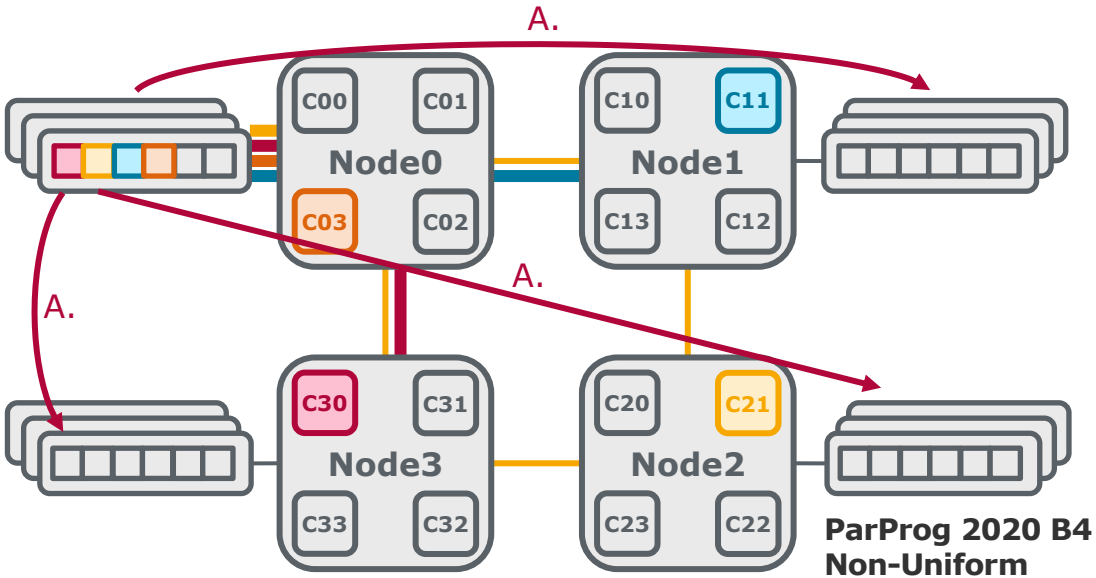


Non-Uniform Memory Access
Data Access Patterns

Multiple tasks on multiple nodes access private buffers on single node

A. Relocate remote buffers to local memory

➢ Reduce memory controller contention

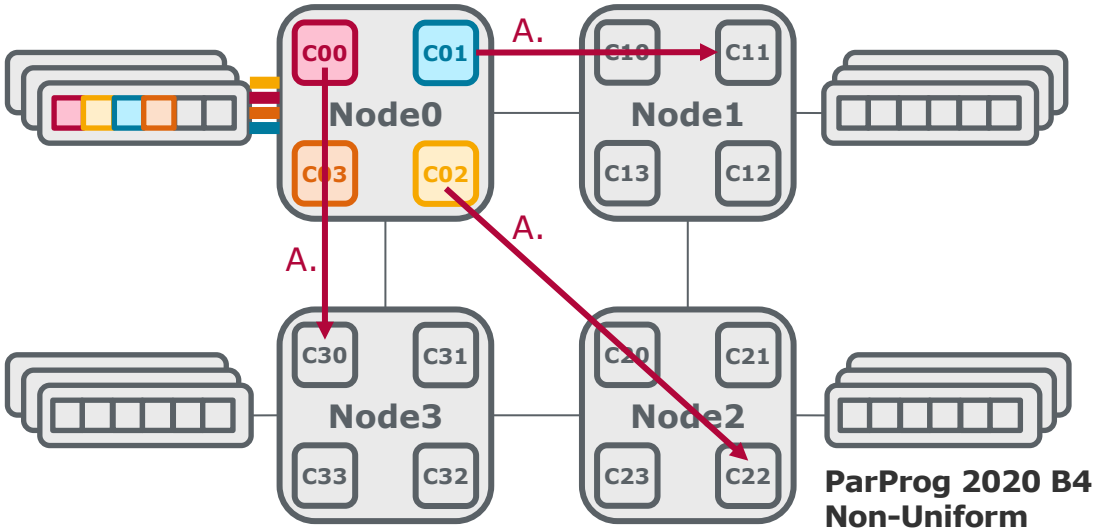


Non-Uniform Memory Access
Data Access Patterns

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

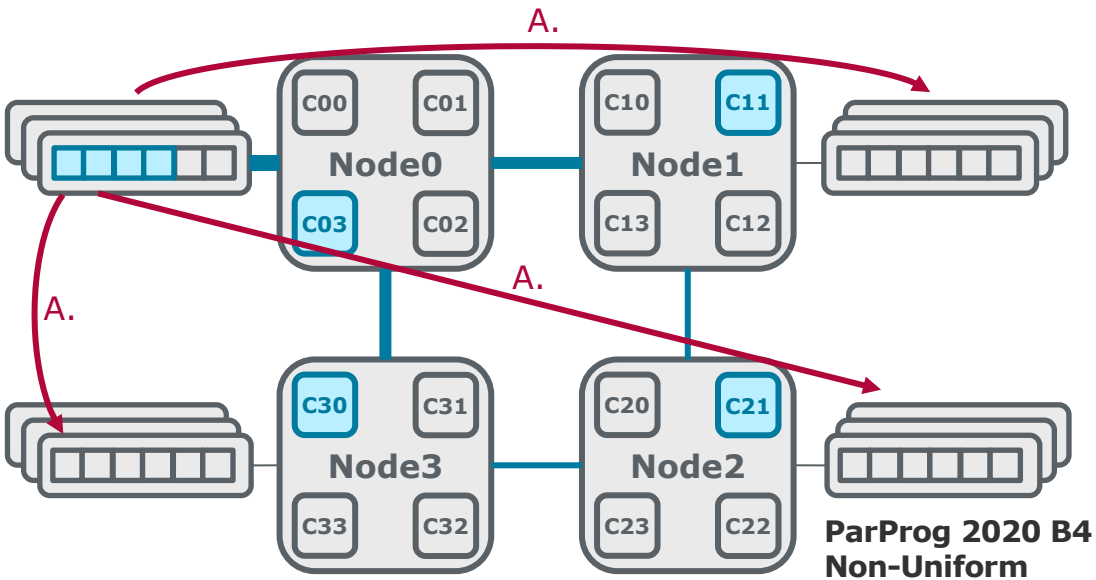


Non-Uniform Memory Access
Data Access Patterns

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

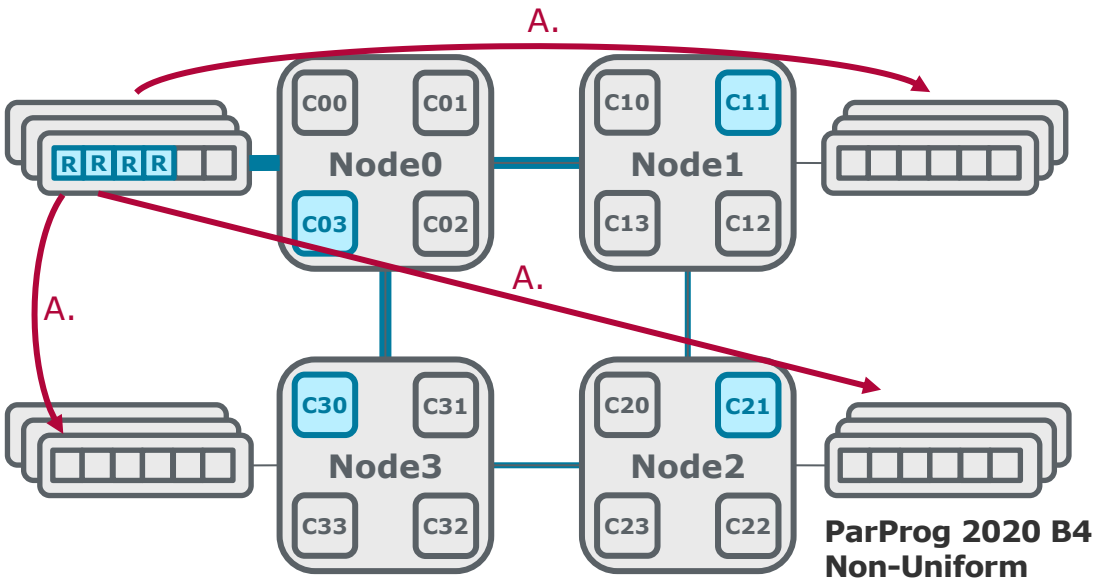


Non-Uniform Memory Access
Data Access Patterns

Tasks on multiple nodes <u>read</u> a shared buffer on single node

A. Read only: Duplicate buffer on every node

➢ Avoid inter node traffic entirely



ParProg 2020 B4 Non-Uniform Memory Access

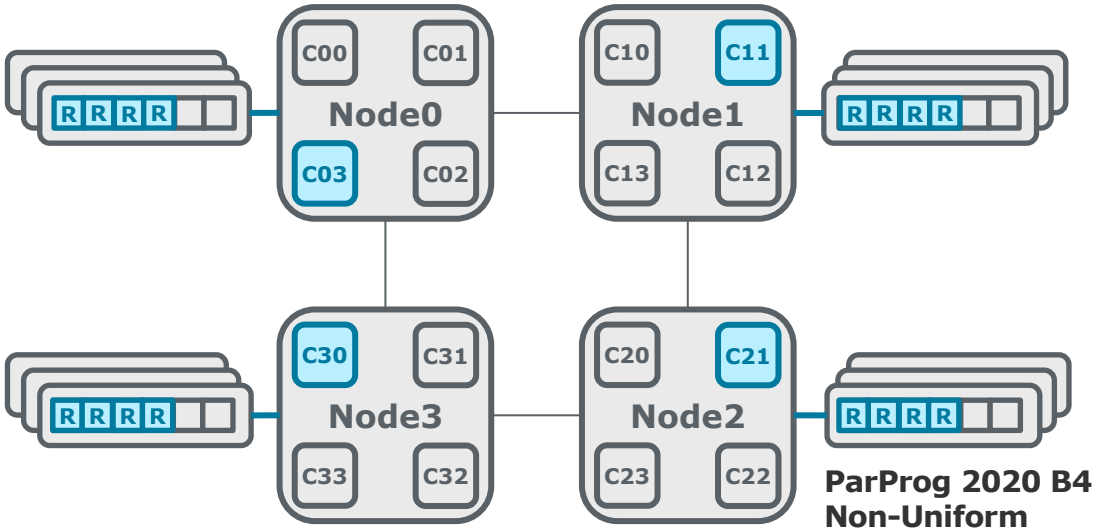
Felix Eberhardt

Non-Uniform Memory Access
Data Access Patterns

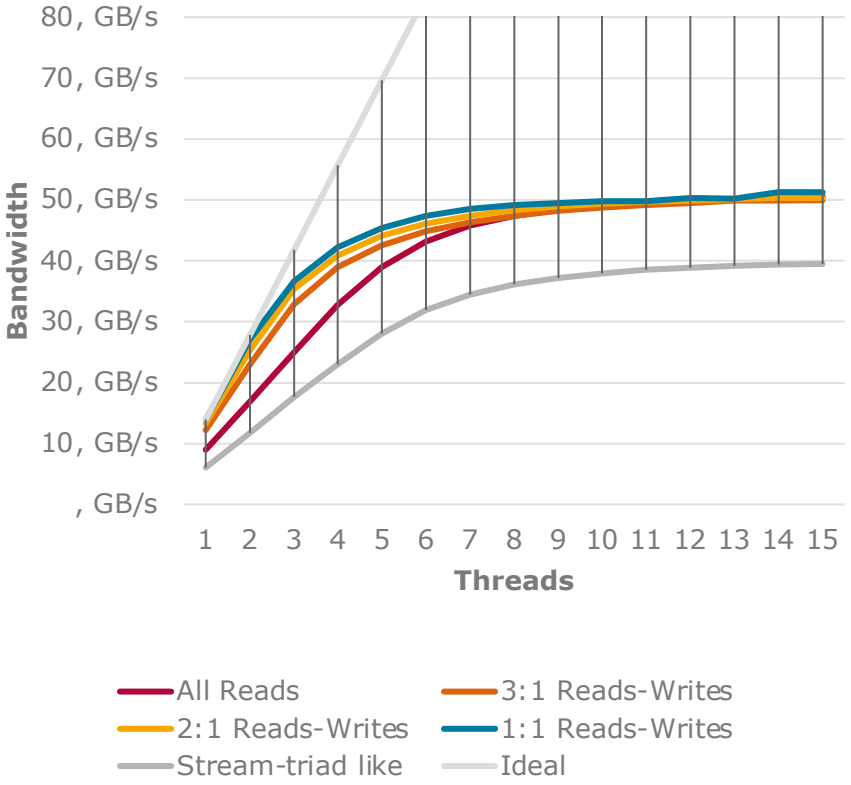
Tasks on multiple nodes <u>read</u> a shared buffer on single node

A. Read only: Duplicate buffer on every node

➢ Avoid inter node traffic entirely



Non-Uniform Memory Access
Local Bandwidth Characteristics

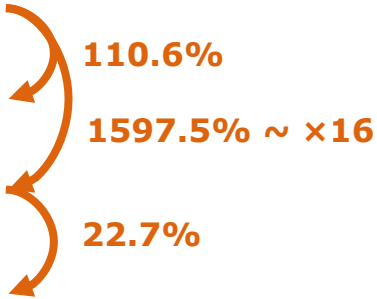


Experiment on SGI UV 300H:
Threads on a single socket generate independent memory traffic

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

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



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

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, ...)

Analzye 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**

Non-Uniform Memory Access
Thread Placement

Tradeoff:

computational load balancing ◇ 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

Hasso Plattner Institut

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
```

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:

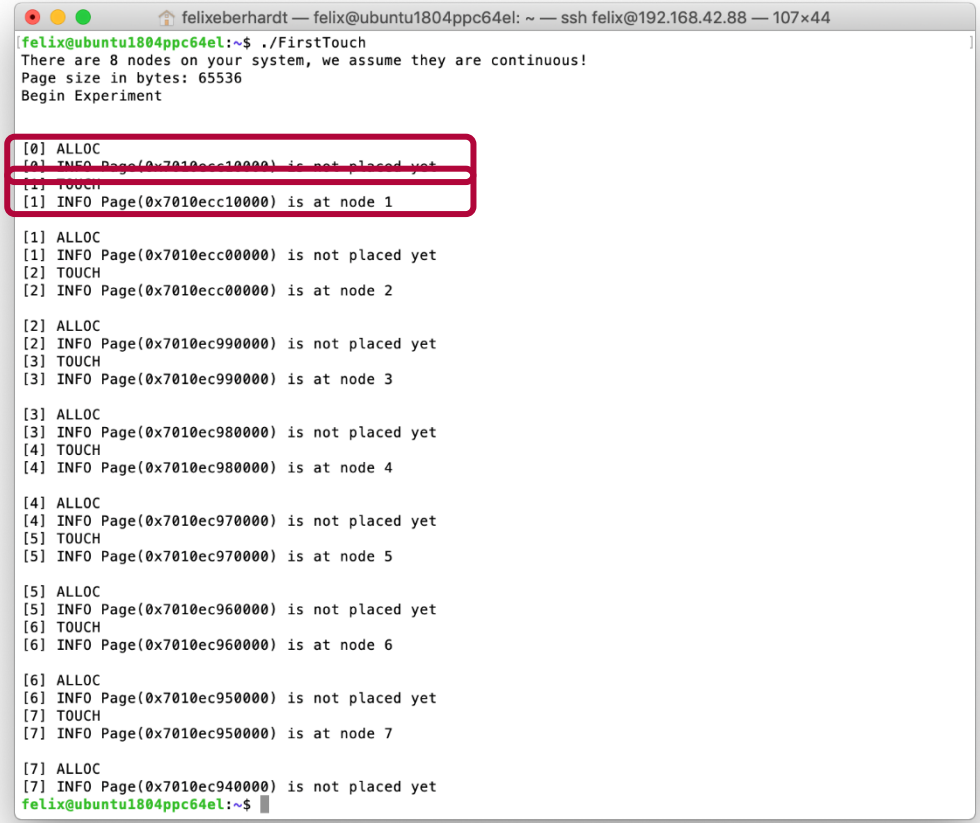
```
move_pages(pid, count, **pages, *nodes, *status, flags);
```

```
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



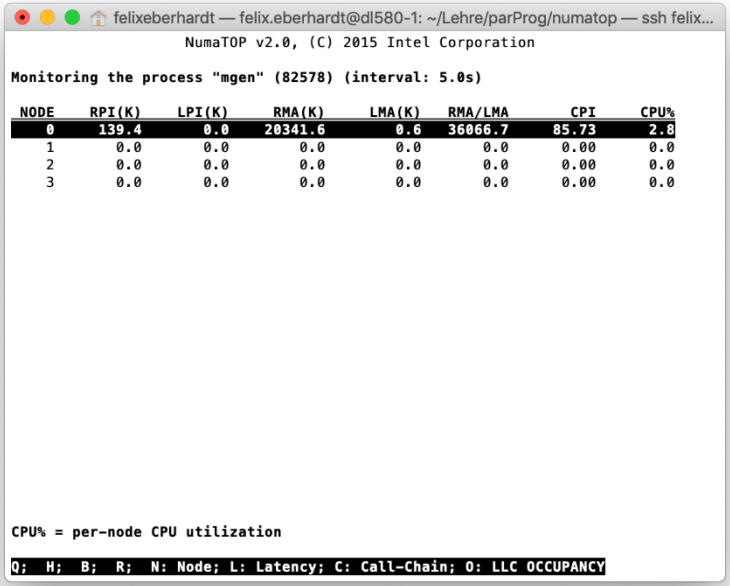
Non-Uniform Memory Access - Toolbox
Topology Discovery

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
- …



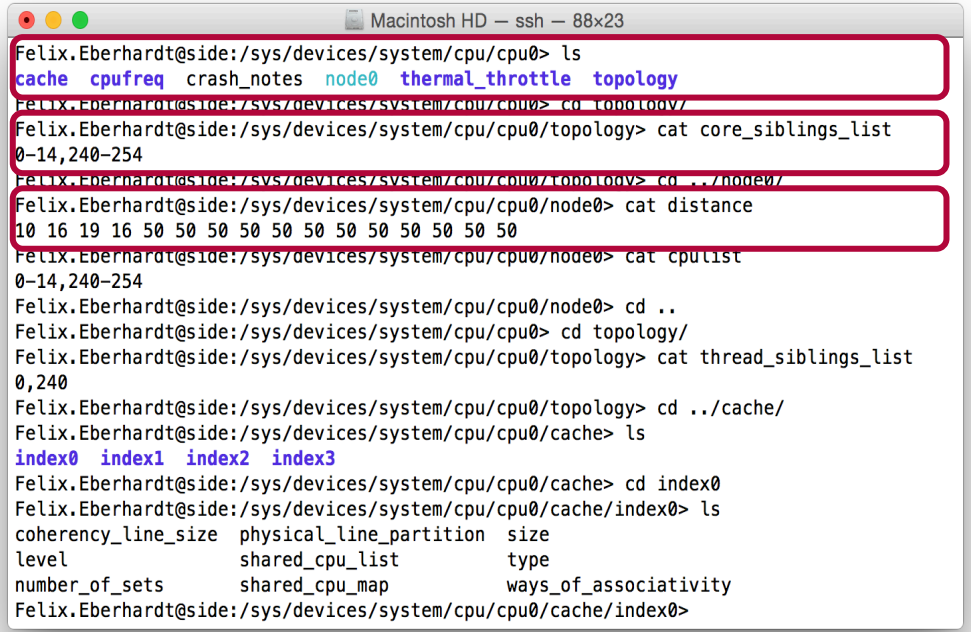
numatop: top focused on NUMA-related information

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



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 — 88×23

```
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:~>
```

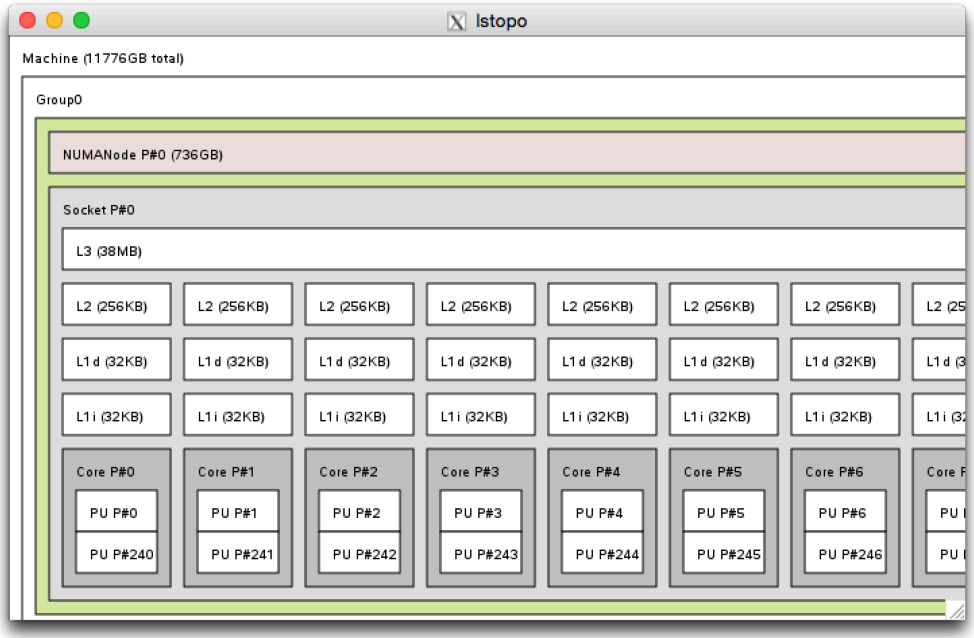
Non-Uniform Memory Access - Toolbox
Topology Discovery: hwloc / lstopo

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



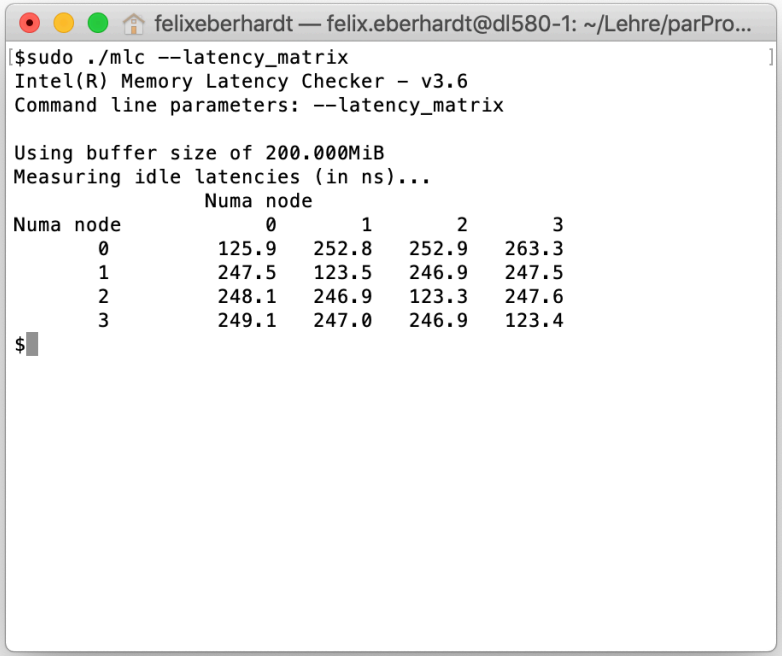
Non-Uniform Memory Access - Toolbox
Topology Discovery: Memory Latency Checker

Empirical information provided:

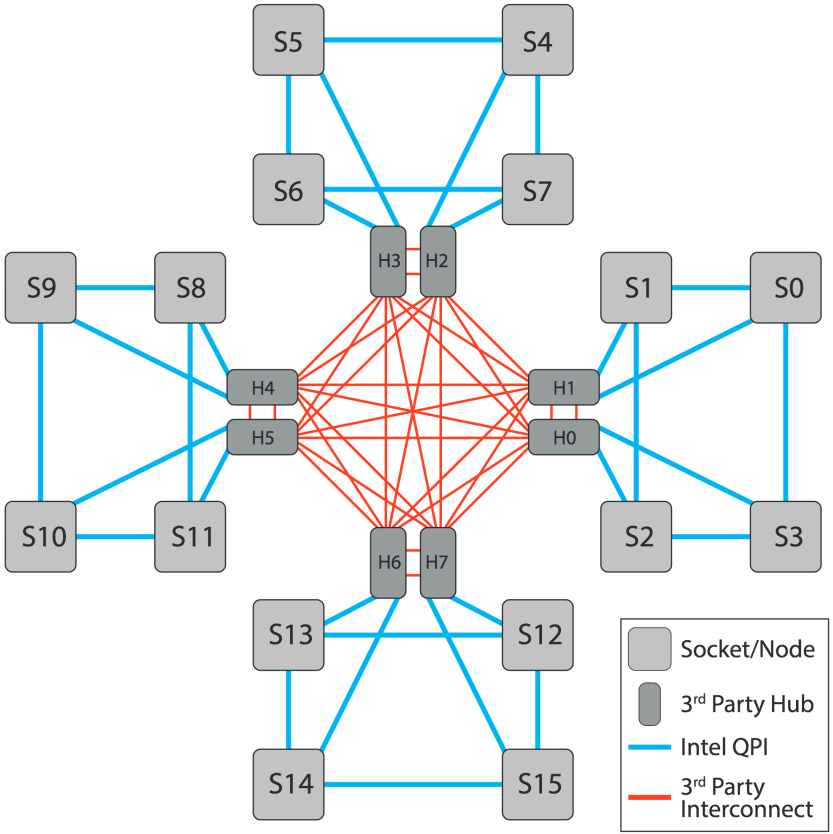
- Latencies to local memory hierarchy
- Bandwidth to local memory hierachy
- Latencies between NUMA nodes
- Bandwidth between NUMA nodes
- Latencies of Cache-to-Cache transfers
- Latencies under load

Restrictions:

- Only on Intel Processors
- No source code available



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



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

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

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

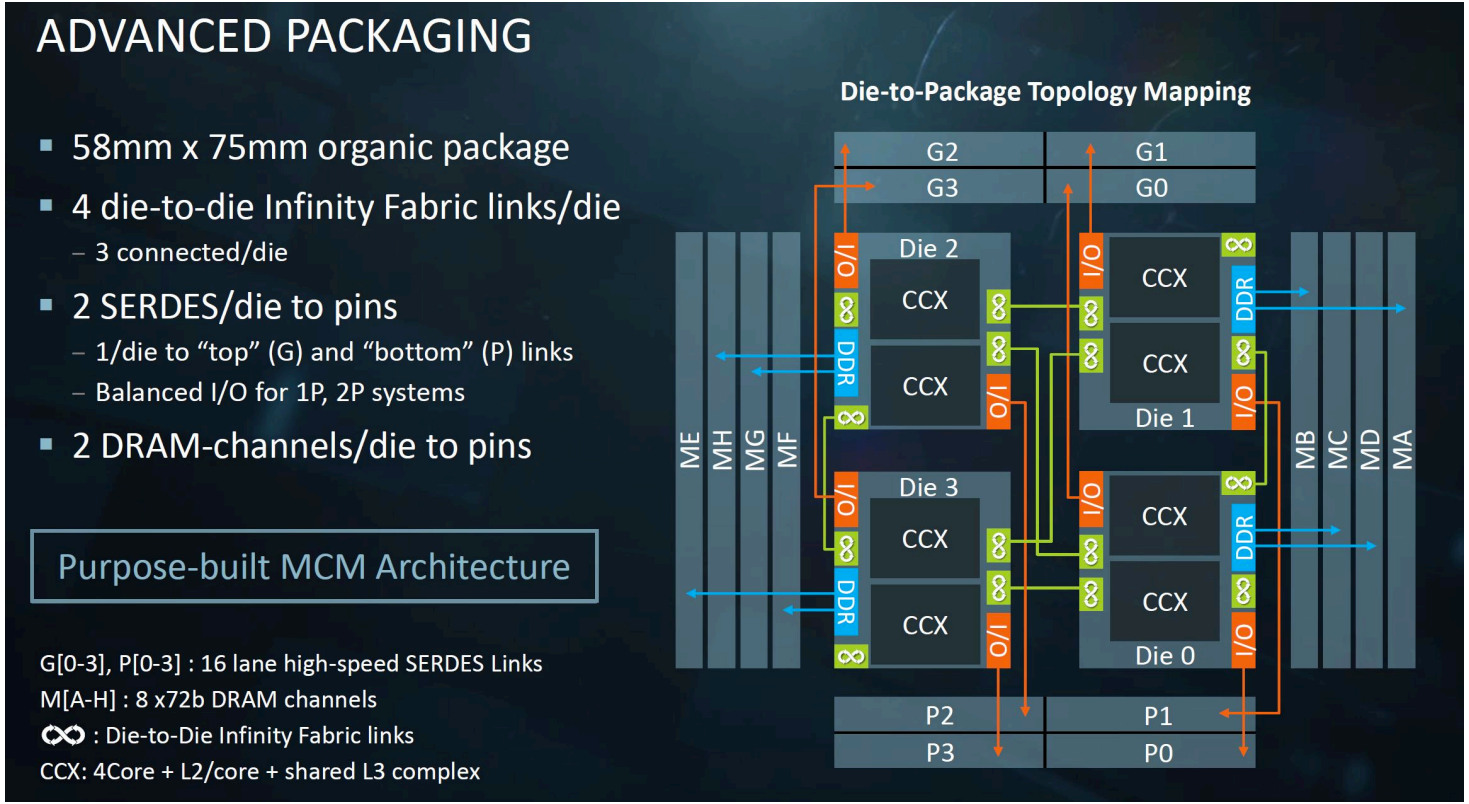
Measured Bandwidth

- 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

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



https://www.servethehome.com/wp-content/uploads/2017/08/AMD-EPYC-Infinity-Fabric-Topology-Mapping.jpg

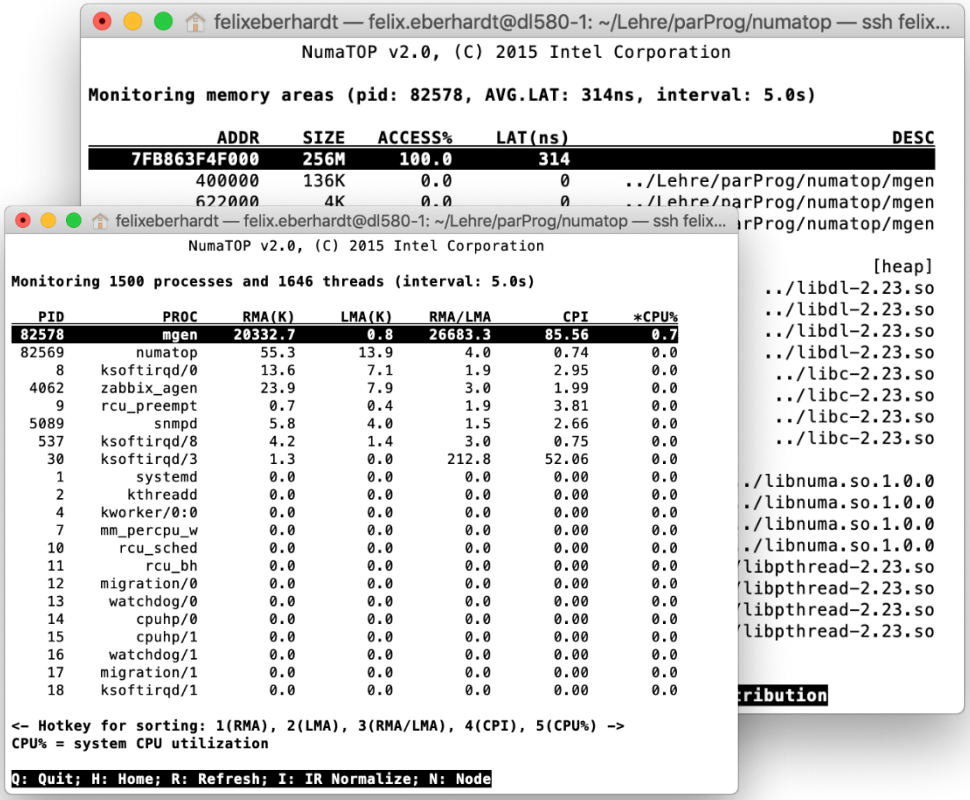
Non-Uniform Memory Access - Toolbox
System Performance: numatop

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



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