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

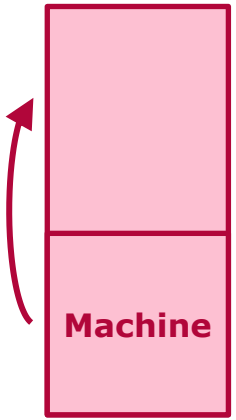
Recap
Optimization Goals

- 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

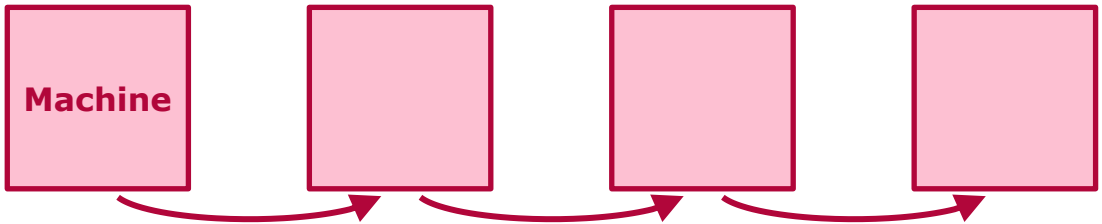
Non-Uniform Memory Access
Context: Scalability

- Two basic approaches to scaling compute 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

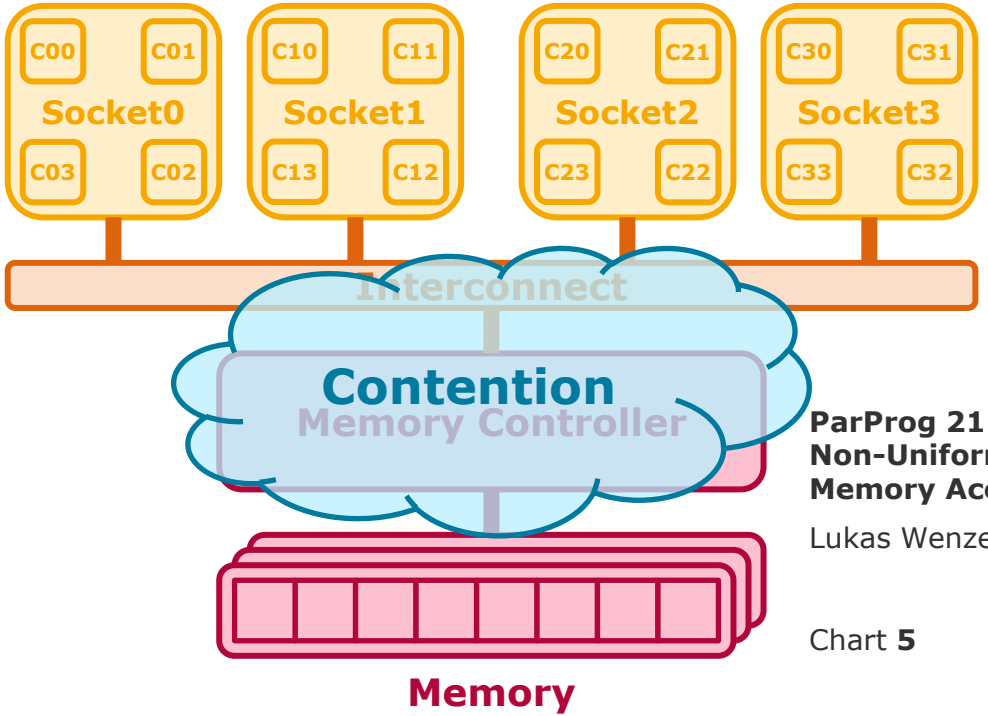
- Two basic approaches to scaling compute hardware:
  - **Scale-Out**: connect more machines in a loosely coupled network
    - User perceives multiple communicating machines in a shared-nothing system



Non-Uniform Memory Access
Context: Uniform Memory Access Machines

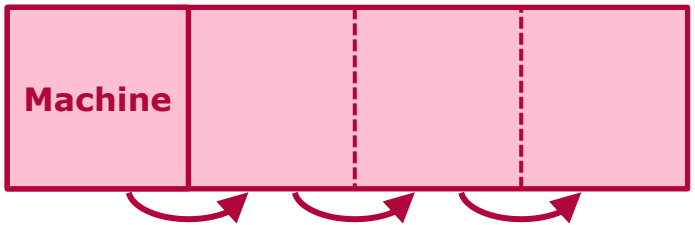
Multiple sockets access main memory through a shared interconnect.

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



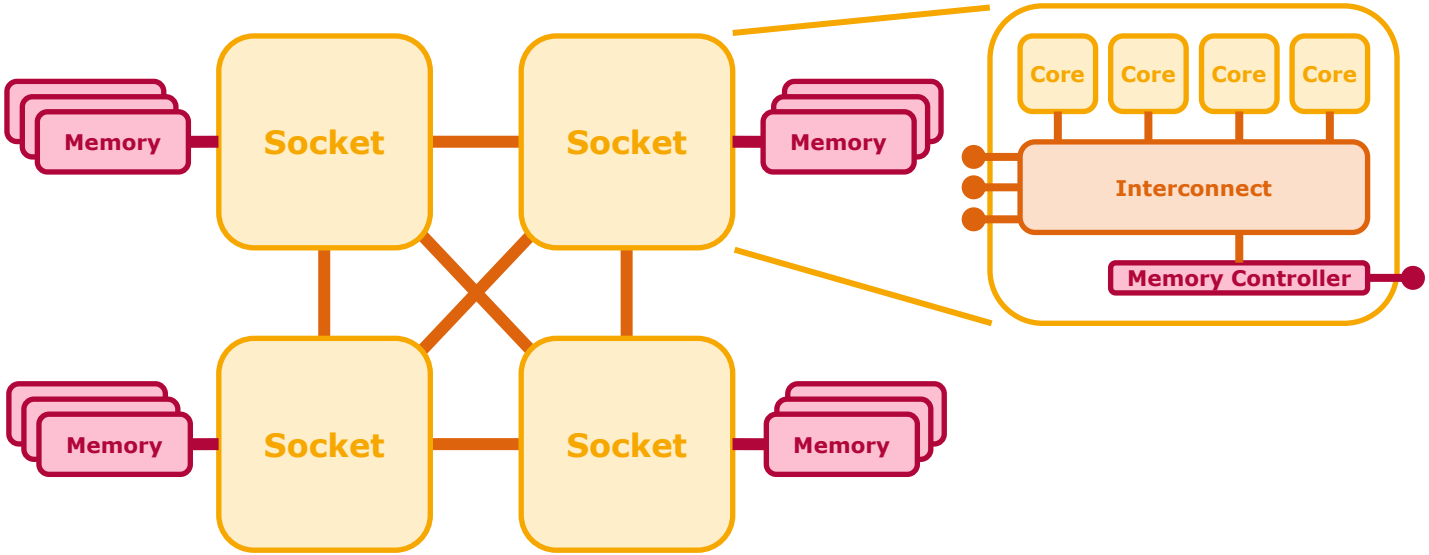
Non-Uniform Memory Access
Context: Scalability

- Recent coherent interconnect technologies enable **hybrid** systems with both scale-up and scale-out characteristics:
  - Collection of self-contained nodes with cores and memory, but interconnect protocols establish shared address space
  - User perceives a shared-memory system, but with the performance characteristics (communication latency and bandwidth) of a distributed-memory system



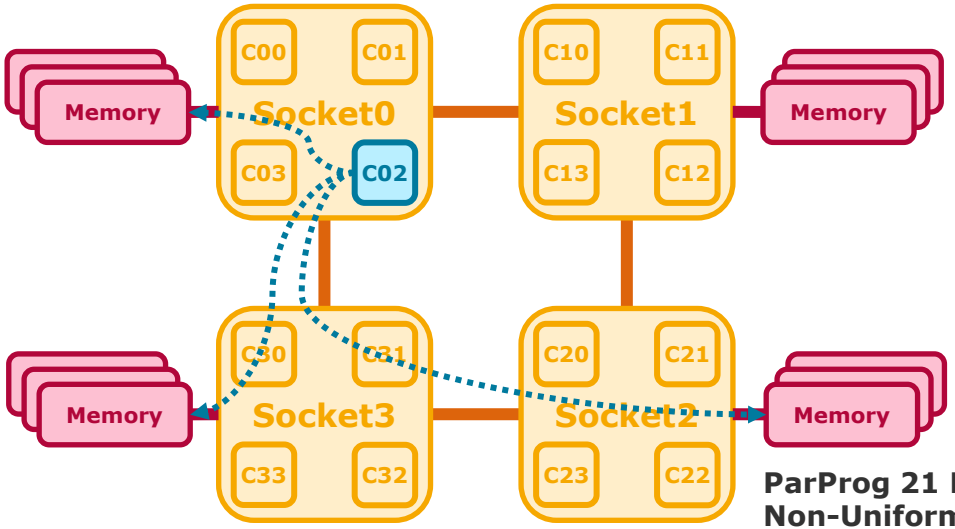
Non-Uniform Memory Access Concept

- 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

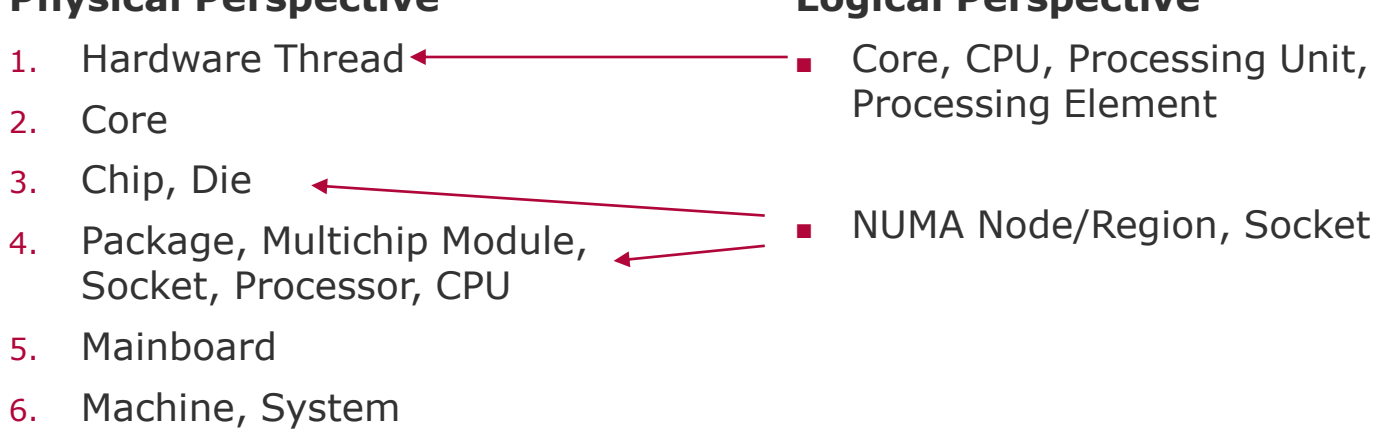
- Multiple **point to point links** between sockets **scale better** than a shared interconnect
- **Multiple memory controllers** partition address space and provide a **higher** total memory **bandwidth**

  (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 may not as wide as on chip connections and memory operations from multiple sockets are handled

- **Predominant architecture for current multi-socket machines**

Non-Uniform Memory Access Terminology

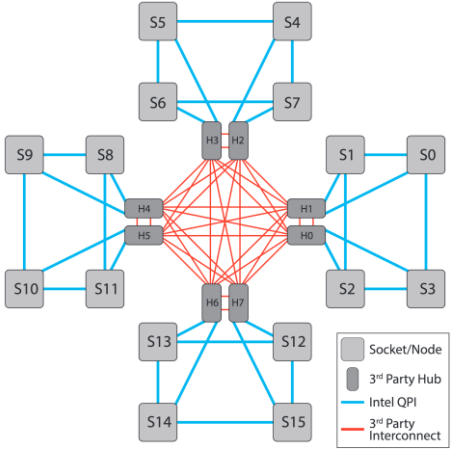


Non-Uniform Memory Access
Example: SGI UV 300H

- 240 Cores (2xSMT)
- 12 TB RAM
- 16 Sockets

What is a Killer Application for such a machine?

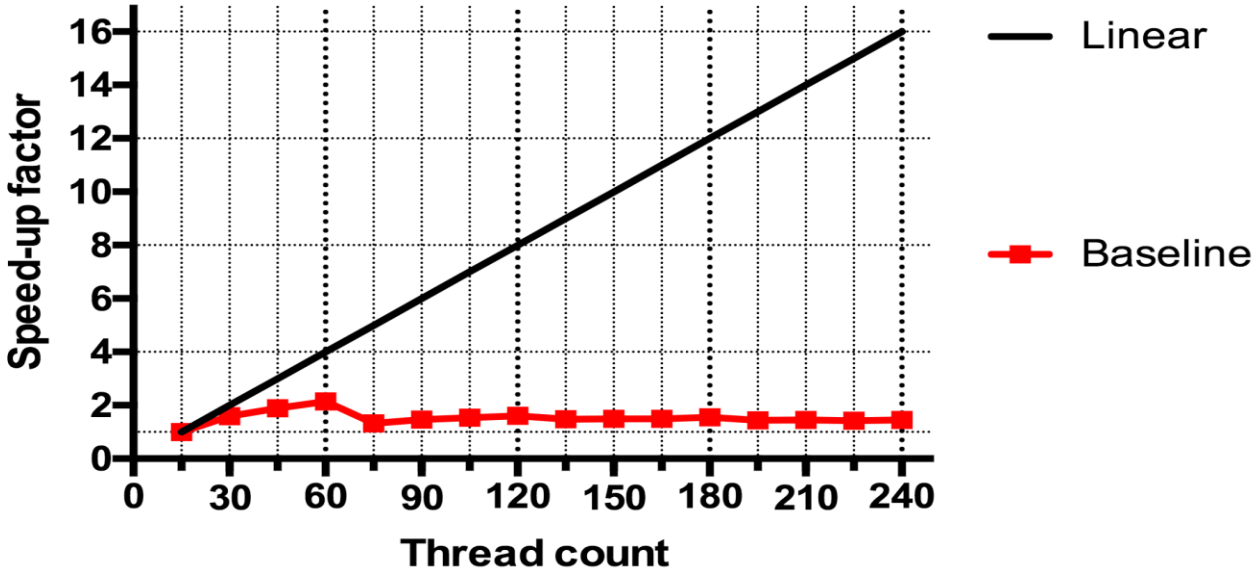
- In-Memory Databases!

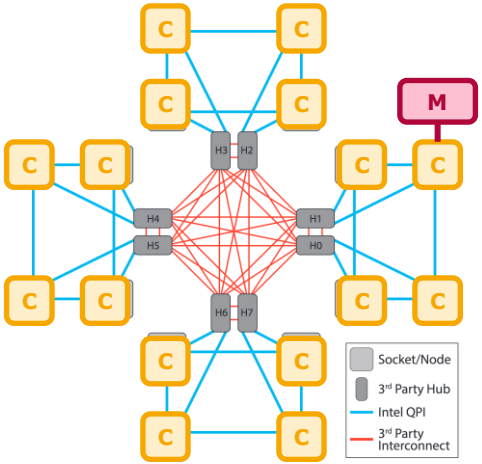


Non-Uniform Memory Access
Example: SGI UV 300H

Experiment: NUMA behavior when scaling a workload

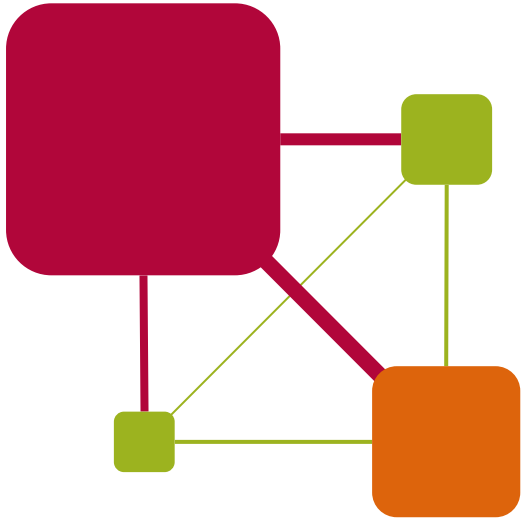
- Machine has 16 sockets x 15 cores x 2-way SMT
  - Threads are placed in locality order, using 1 hardware thread per core
- ➢ Performance **degrades** when using more than two sockets!

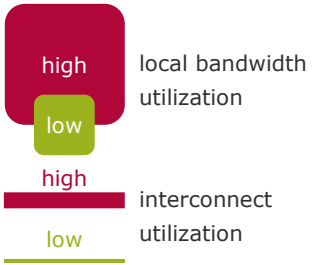




Non-Uniform Memory Access Characteristics

- 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





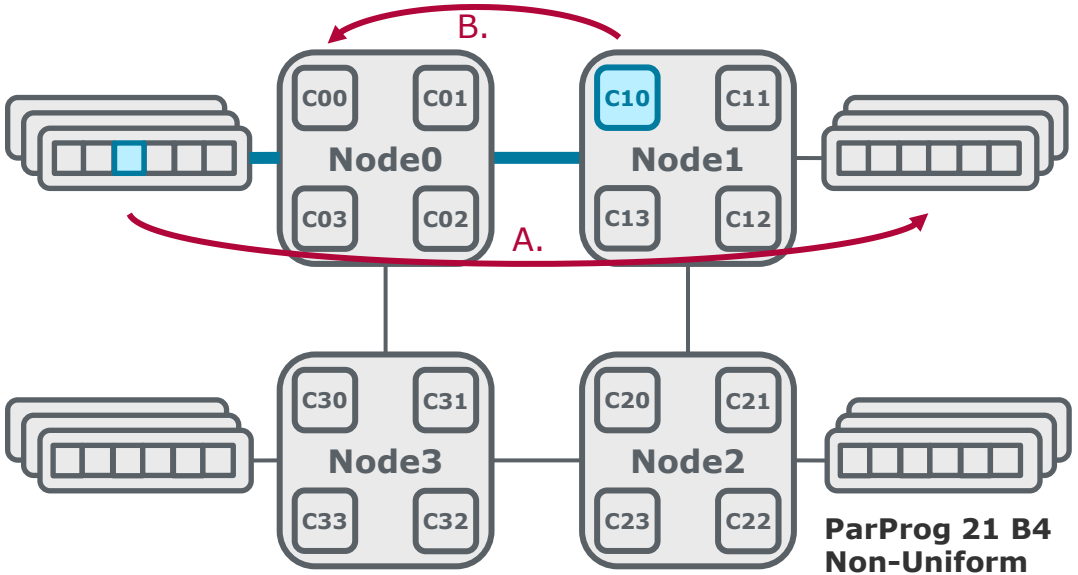
Non-Uniform Memory Access
Data Access Patterns

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

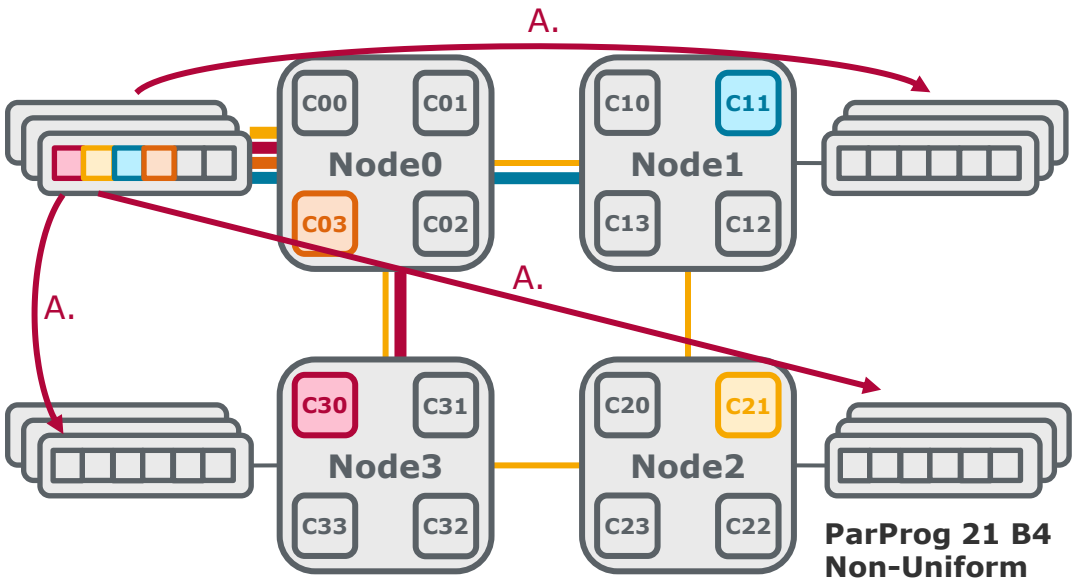


Non-Uniform Memory Access
Data Access Patterns

Multiple tasks on multiple nodes access private buffers on single node

A. Relocate remote buffers to local memory

➢ Reduce memory controller contention



ParProg 21 B4 Non-Uniform Memory Access

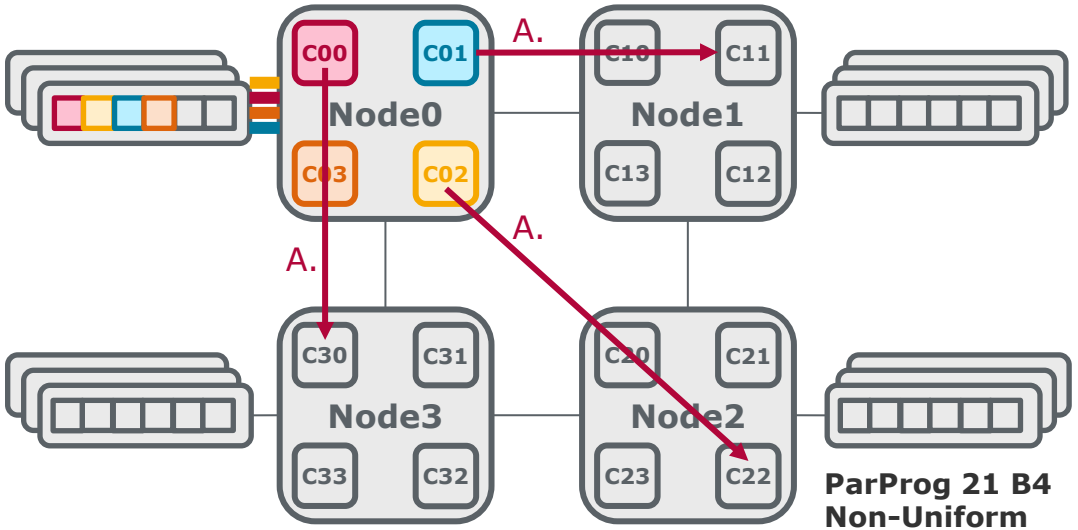
Lukas Wenzel

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

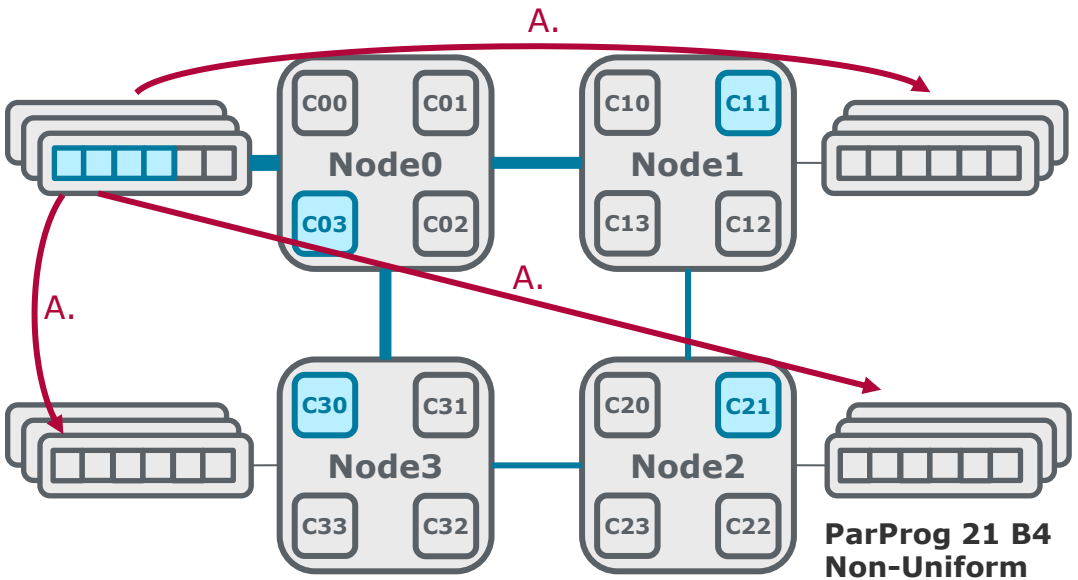


Non-Uniform Memory Access
Data Access Patterns

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

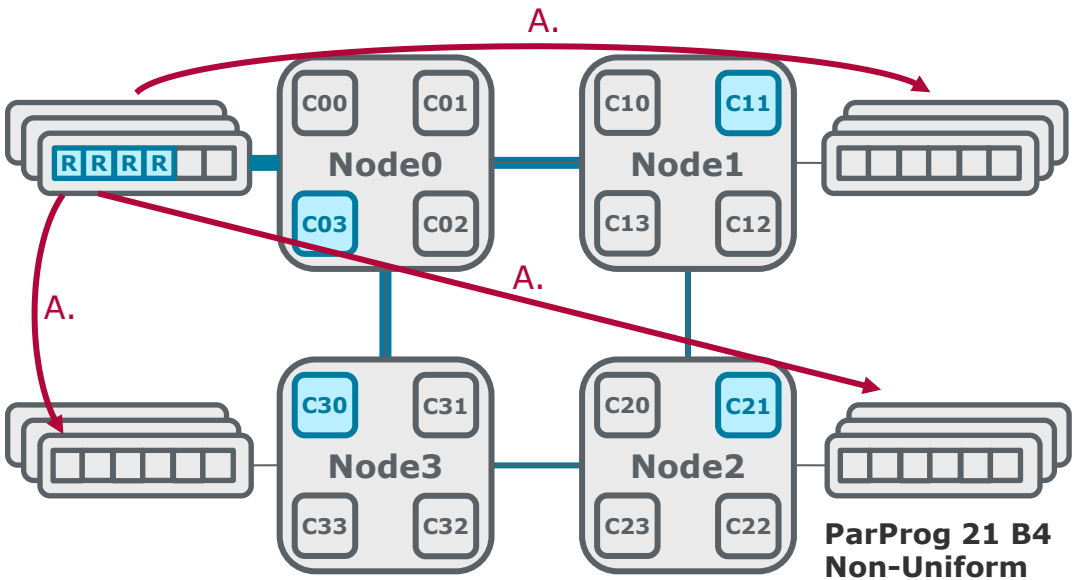


Non-Uniform Memory Access
Data Access Patterns

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

A. Read only: Duplicate buffer on every node

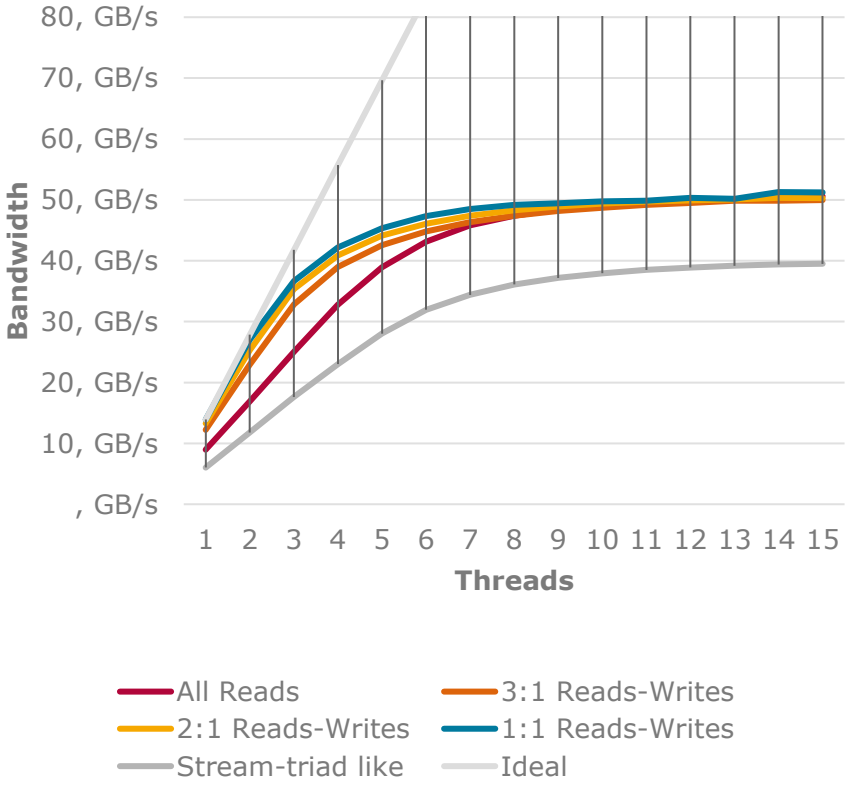
➢ Avoid inter node traffic entirely



ParProg 21 B4 Non-Uniform Memory Access

Lukas Wenzel

Non-Uniform Memory Access
Local Bandwidth Characteristics

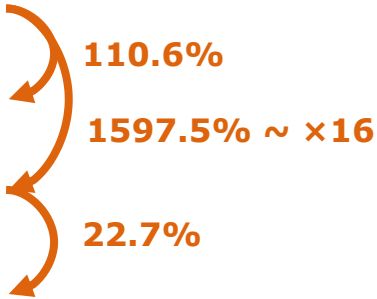


Experiment on SGI UV 300H:
Threads on a single socket generate local memory traffic

- 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 hardware thread a (software) thread can be scheduled on

  - **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**: Placement is fixed at allocation either by policy or programmer request
  - First-touch policy (defacto standard)
  - Fixed on a single node
  - Interleaved between multiple nodes
  - *Replicated on multiple nodes (consistency!)*
- **Dynamic**: Page frames can migrate between different nodes after allocation

  - ***Data follows computation***

Non-Uniform Memory Access - Toolbox
External Placement Control

System utilities to control placement decisions of a process from outside.

`numactl` comes with `libnuma`

```
> numactl --cpunodebind=1 --membind=1 ./workload
```

- **Thread Placement Options**

  Set default affinity mask for threads in the new process.
  - `--physcpubind=<cpus>` Pins threads to individual hardware threads
  - `--cpunodebind=<nodes>` Pins threads on a NUMA-node granularity
- **Data Placement**

  Set the page allocation policy for the new process.
  - `--interleave=<nodes>` Interleaved across the given set of nodes
  - `--membind=<nodes>` Static placement within the given node or set of nodes
- Arguments are comma separated lists of cpu/node ids or ranges

  e.g. `--physcpubind=0,2,4,6,16-31`

taskset

alternative tool to control affinity masks in already running process

Non-Uniform Memory Access - Toolbox
Internal Placement Control

APIs to control placement from within a running process.

Thread Placement

- Linux-Systemcall:

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

- Using Pthread:

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

GNU extension

- Using libnuma:

```
int numa_run_on_node(int node)
int numa_run_on_node_mask(struct bitmask * nodemask)
```

libnuma

```
> man 3 numa
```

Non-Uniform Memory Access - Toolbox
Internal Placement Control

APIs to control placement from within a running process.

Data Placement

- Using libnuma:
  - Allocation

```
void * numa_alloc_onnode(size_t size, int node)
void * numa_alloc_interleaved_subset(size_t size, struct bitmask * mask)
```

  - Relocation

```
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
Topology Discovery

System utilities to discover and analyze the NUMA topology.

- Inspecting Linux `sysfs`:

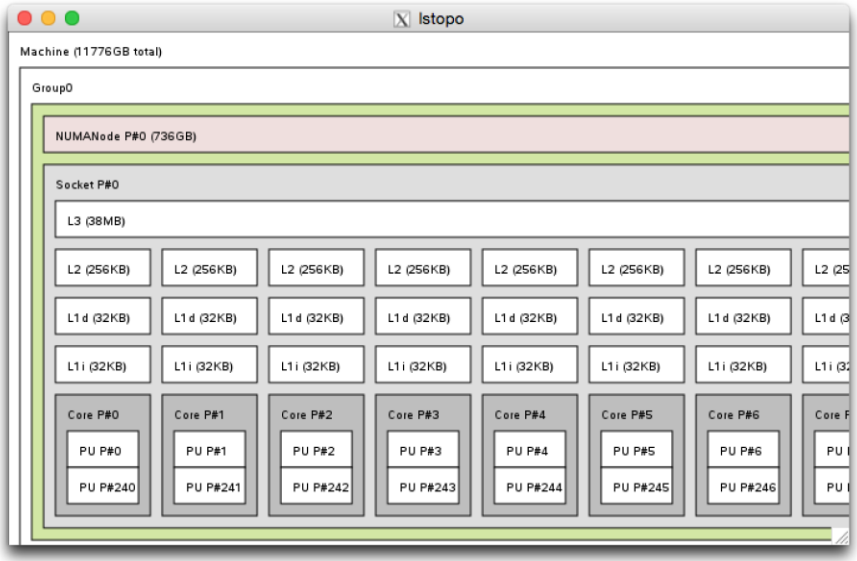
  ```
  > ls /sys/devices/system/node/
  ```

- Using `numactl`:

  ```
  > numactl --hardware
  ```

- Using `hwloc`

  ```
  > lstopo
  ```



Intel **`numatop`**

tool to analyze bandwidth and latency as observed by a running process

Non-Uniform Memory Access - Toolbox
Topology Discovery

APIs to query the NUMA environment of a running process.

- Using **`libnuma:`**
  - `int numa_num_configured_nodes()`
    `int numa_num_configured_cpus()`
    get the total number of NUMA nodes or hw-threads in the system
  - `int numa_distance(int node1, int node2)`
    get the distance between two NUMA nodes as reported by ACPI
  - `int numa_node_to_cpus(int node, struct bitmask * mask)`
    `int numa_node_of_cpu(int cpu)`
    associate NUMA nodes and hw-threads
  - `long numa_node_size(int node, long * free)`
    get the total and free memory on a NUMA node

Non-Uniform Memory Access - Toolbox
Topology Discovery

System utilities to discover and analyze the NUMA topology.

- **Intel Memory Latency Checker**

  ```
  > ./mlc --latency_matrix --bandwidth_matrix
  ```

  Performs benchmarks to measure:
  - Latency to local memory hierarchy
  - Bandwidth to local memory hierachy
  - Latencies between NUMA nodes
  - Bandwidths between NUMA nodes
  - Latencies of Cache-to-Cache transfers

  Restrictions:
  - Only on Intel Processors
  - No source code available

```
$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
$
```

Non-Uniform Memory Access - Toolbox Experiments: IBM Power System E880

1. Explore topology of an IBM Power System E880
   - 2 Blades × 4 Sockets × 6 Cores × 8 SMT
2. Read topology information with `libnuma`
3. Experiment with First-Touch policy

```
lwenzel@ubuntu1804ppc64el : /home/lwenzel/numa
2021-05-16 15:43:50 > ./numatest
Pagesize is 64 KiB

Moved to cpu 0 on node 0.
Page 0x75fac0ee0000 is not placed yet
Moved to cpu 0 on node 0.
Page 0x75fac0ee0000 accessed from node 0
Page 0x75fac0ee0000 is at node 0
Moved to cpu 0 on node 0.
Page 0x75fac0ee0000 accessed from node 0
Page 0x75fac0ee0000 is at node 0

Moved to cpu 48 on node 1.
Page 0x75fac0ea0000 is not placed yet
Moved to cpu 96 on node 2.
Page 0x75fac0ea0000 accessed from node 2
Page 0x75fac0ea0000 is at node 2
Moved to cpu 144 on node 3.
Page 0x75fac0ea0000 accessed from node 3
Page 0x75fac0ea0000 is at node 2

Moved to cpu 96 on node 2.
Page 0x75fac0e60000 is not placed yet
Moved to cpu 192 on node 4.
Page 0x75fac0e60000 accessed from node 4
Page 0x75fac0e60000 is at node 4
Moved to cpu 288 on node 6.
Page 0x75fac0e60000 accessed from node 6
Page 0x75fac0e60000 is at node 4

Moved to cpu 144 on node 3.
Page 0x75fac0e20000 is not placed yet
Moved to cpu 288 on node 6.
Page 0x75fac0e20000 accessed from node 6
Page 0x75fac0e20000 is at node 6
Moved to cpu 48 on node 1.
Page 0x75fac0e20000 accessed from node 1
Page 0x75fac0e20000 is at node 6
```

Non-Uniform Memory Access Experiments: SGI UV-300H

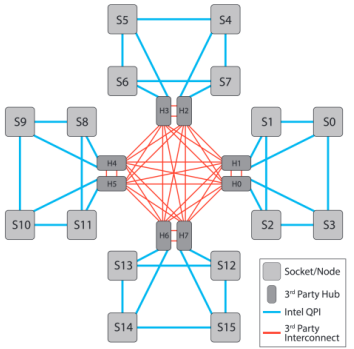
```
> numactl --hardware
```

Logical "distance" between sockets in system topology.

- Opaque measure but related to both latency and bandwidth characteristics.

ACPI Distance Values

10	16	19	16	50	50	50	50	50	50	50	50	50	50	50	50
16	10	16	19	50	50	50	50	50	50	50	50	50	50	50	50
19	16	10	16	50	50	50	50	50	50	50	50	50	50	50	50
16	19	16	10	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	10	16	19	16	50	50	50	50	50	50	50	50
50	50	50	50	16	10	16	19	50	50	50	50	50	50	50	50
50	50	50	50	19	16	10	16	50	50	50	50	50	50	50	50
50	50	50	50	16	19	16	10	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	10	16	19	16	50	50	50	50
50	50	50	50	50	50	50	50	16	10	16	19	50	50	50	50
50	50	50	50	50	50	50	50	19	16	10	16	50	50	50	50
50	50	50	50	50	50	50	50	16	19	16	10	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	10	16	19	16
50	50	50	50	50	50	50	50	50	50	50	50	16	10	16	19
50	50	50	50	50	50	50	50	50	50	50	50	19	16	10	16
50	50	50	50	50	50	50	50	50	50	50	50	16	19	16	10



ParProg 21 B4 Non-Uniform Memory Access

Lukas Wenzel

Non-Uniform Memory Access Experiments: SGI UV-300H

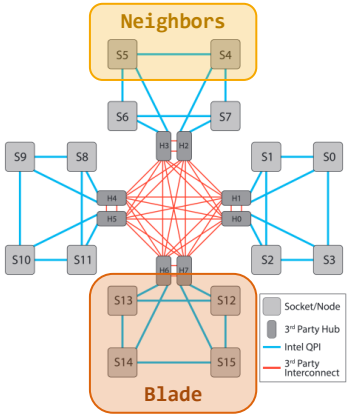
```
> ./mlc --latency-matrix
```

- Four distinct latency classes:

Local	110 ns	×1
Neighbor	200 ns	×1.8
Blade	230 ns	×2.1
Far Remote	480 ns	×4.4

Memory Latency in ns

111	197	231	204	482	484	481	485	485	489	487	489	488	489	488	488
203	111	191	234	481	481	480	482	482	486	486	488	486	489	487	488
242	197	112	197	482	484	480	484	486	488	485	488	488	488	487	488
212	233	189	113	481	481	480	482	484	488	483	488	486	488	487	488
481	482	480	483	114	194	225	200	480	481	481	487	483	489	487	488
481	482	480	483	197	112	192	232	480	481	481	487	483	489	487	488
482	484	481	484	234	195	111	192	480	483	481	488	487	489	486	487
481	483	480	482	203	230	190	113	480	481	480	488	485	489	485	486
487	488	484	488	481	481	480	484	114	195	226	200	481	489	482	483
485	487	483	488	480	481	480	482	191	114	190	231	481	487	481	482
487	488	481	487	482	482	480	482	227	195	114	195	481	488	481	482
488	488	483	488	482	483	480	483	198	229	189	114	401	488	482	483
488	490	486	489	485	488	487	488	481	485	483	488	112	193	227	200
487	488	484	488	484	488	485	489	480	483	482	488	190	114	191	230
487	489	485	489	485	489	483	489	481	485	482	488	227	194	112	197
487	488	484	488	489	488	483	488	480	485	481	487	195	227	190	113



ParProg 21 B4 Non-Uniform Memory Access

Lukas Wenzel

Non-Uniform Memory Access Experiments: SGI UV-300H

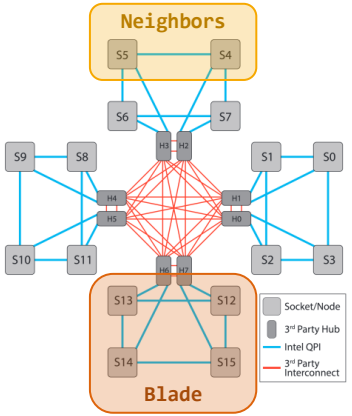
```
> ./mlc --bandwidth-matrix
```

- Non-local bandwidths are all similar:

Local	51.4 GB/s	×5.4
Neighbor	12.5 GB/s	×1.1
Blade	11.5 GB/s	×1.0
Far Remote	11.3 GB/s	×1

Memory Bandwidths in GB/s

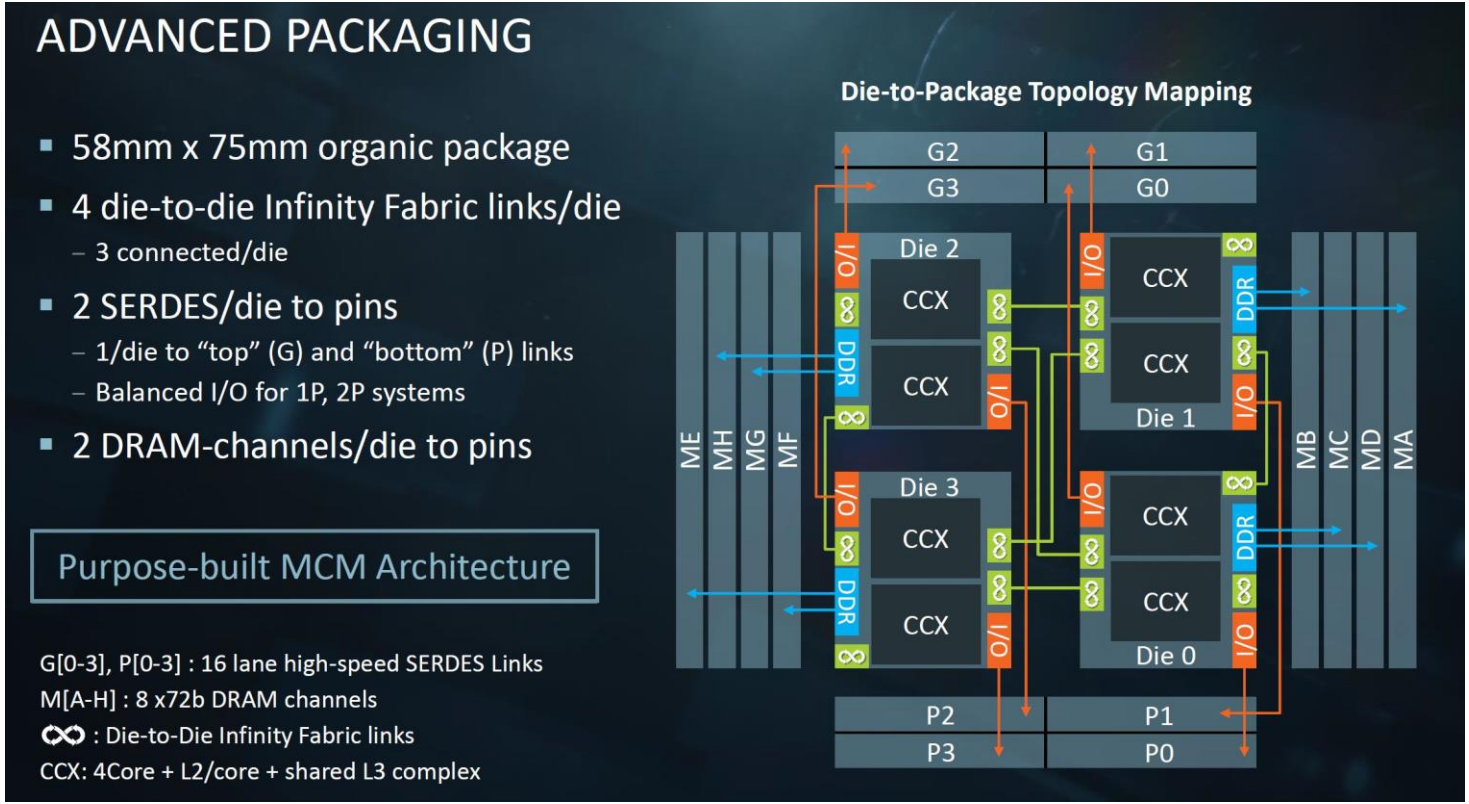
51,3	12,8	11,7	12,1	11,4	11,4	11,3	11,4	11,3	11,4	11,4	11,2	11,1	11,2	11,3	11,3
12,7	51,7	12,1	11,5	11,4	11,4	11,3	11,4	11,3	11,3	11,4	11,2	11,1	11,2	11,3	11,3
12,8	12,1	51,8	12,7	11,4	11,4	11,3	11,4	11,3	11,4	11,4	11,3	11,1	11,2	11,3	11,3
12,1	11,5	12,7	51,3	11,4	11,4	11,3	11,4	11,3	11,3	11,4	11,2	11,1	11,2	11,3	11,3
11,1	11,4	11,2	11,4	51,3	12,8	11,7	12,1	11,3	11,4	11,4	11,2	11,1	11,2	11,3	11,3
11,1	11,4	11,3	11,4	12,7	51,7	12,1	11,5	11,3	11,4	11,4	11,2	11,1	11,2	11,3	11,3
11,1	11,4	11,3	11,4	12,1	12,1	51,7	12,7	11,3	11,4	11,4	11,2	11,1	11,2	11,3	11,3
11,1	11,4	11,2	11,4	12,1	11,5	12,8	51,6	11,3	11,4	11,4	11,2	11,1	11,2	11,3	11,3
11,1	11,4	11,2	11,4	11,4	11,4	11,3	11,4	51,3	12,7	11,7	12,1	11,1	11,2	11,3	11,3
11,1	11,4	11,2	11,4	11,4	11,4	11,3	11,4	12,8	51,3	12,1	11,5	11,1	11,2	11,3	11,3
11,1	11,4	11,2	11,4	11,4	11,4	11,3	11,4	12,1	12,1	51,3	12,7	11,1	11,2	11,3	11,3
11,1	11,4	11,2	11,4	11,4	11,4	11,3	11,4	12,1	11,5	12,8	51,3	11,1	11,2	11,3	11,3
11,1	11,4	11,2	11,4	11,4	11,4	11,3	11,4	11,3	11,4	11,4	11,2	51,8	12,7	11,7	12,1
11,1	11,4	11,2	11,4	11,4	11,4	11,3	11,4	11,3	11,4	11,4	11,2	12,8	51,2	12,1	11,5
11,1	11,4	11,2	11,4	11,4	11,4	11,3	11,4	11,3	11,3	11,4	11,3	12,1	12,1	51,4	12,8
11,1	11,4	11,2	11,4	11,4	11,4	11,2	11,4	11,3	11,3	11,4	11,3	12,1	11,5	12,8	51,4



ParProg 21 B4 Non-Uniform Memory Access

Lukas Wenzel

Non-Uniform Memory Access
Outlook: Single Socket NUMA



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

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

And now for a break and a bowl of Hōjicha.

*or beverage of your choice