ACELab Technical Report TR-2021-01

Mapping Core and L3 Slice Numbering to Die Location in Intel Xeon Scalable Processors

Document Revision 1.0
December 10, 2020, revised to 2021-01-13
Status: Release

John D. McCalpin
mccalpin@tacc.utexas.edu
Advanced Computing Evaluation Laboratory
Texas Advanced Computing Center
The University of Texas at Austin
www.tacc.utexas.edu

The University of Texas at Austin and the Texas Advanced Computing Center make no warranty, express or implied, nor assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed.

Mapping Core and L3 Slice Numbering to Die Location in Intel Xeon Scalable Processors

John D. McCalpin mccalpin@tacc.utexas.edu
Texas Advanced Computing Center
The University of Texas at Austin

Abstract: A methodology for mapping from user-visible core and L3 slice numbers to locations on the processor die is presented, along with results obtained from systems with Intel Xeon Scalable Processors ("Skylake Xeon" and "Cascade Lake Xeon") at the Texas Advanced Computing Center. The current methodology is based on the data traffic counters in the 2-D mesh on-chip-network, with the measurements revealing unexpected and counterintuitive transformations of the meanings of "left" and "right" in different regions of the chip. Results show that the numbering of L3 slices is consistent across processor models, while the numbering of cores displays a small number of different patterns, depending on processor model and system vendor.

1 Introduction

The high-level layout of the various elements (core, L3 cache, etc.) of the two-dimensional mesh architecture of the Intel Xeon Scalable Processor Family is documented in a variety of publications, using diagrams such as Figure 1 (from [1]). The diagram clearly shows the two-dimensional mesh structure along with the locations of the cores, the ``CHA/SF/LLC" blocks (described in more detail later), the memory controllers, the UPI interconnect, and the PCIe interfaces.

Curiously, however, Intel provides no information about how the user-visible numbering for the cores, CHA/SF/LLC blocks, and memory controllers corresponds to physical locations on the processor die. Knowing the locations is essential for visualization of activity on the chip and for a variety of detailed performance studies relating to on-chip interconnects, cache coherence, thermal effects, and the trade-offs between locality and scalability in multicore/manycore processors.

This report describes a methodology for deriving the physical locations of each core number and each L3 slice number and reports the results for Intel processors in the TACC Frontera and Stampede2 systems. The approach employs the data traffic counters in the two-dimensional mesh of the processor, with a process pinned to a specific logical processor while reading data from one or both memory controllers. One might anticipate that inverting the data to recover the spatial layout would be straightforward, but the measurements were inconsistent with expected patterns. Eventually it was recognized that an entirely unexpected set of transformations were required to bring the measurements and expectations into agreement – allowing automated or semi-automated deduction of the locations of the cores and L3 slices along with full mapping of the data traffic on the mesh.

2 Background

In a previous report [2], we reviewed the most common patterns of logical processor numbering and how these are related to xAPIC identifiers in recent Intel processor systems. One conclusion of that report is that neither of these unique identifiers has a fixed relationship with the location of the corresponding physical core on the processor die of "mainstream" Intel Xeon processors – *i.e.*, there is at least one layer of indirection between the lowest-level user-visible hardware identifiers (the xAPIC IDs) and the actual hardware.

In the absence of documentation, experimental determination of the locations of the user-visible devices is the only way forward, but the methodology and the interpretation must take extra care to specify all assumptions made.

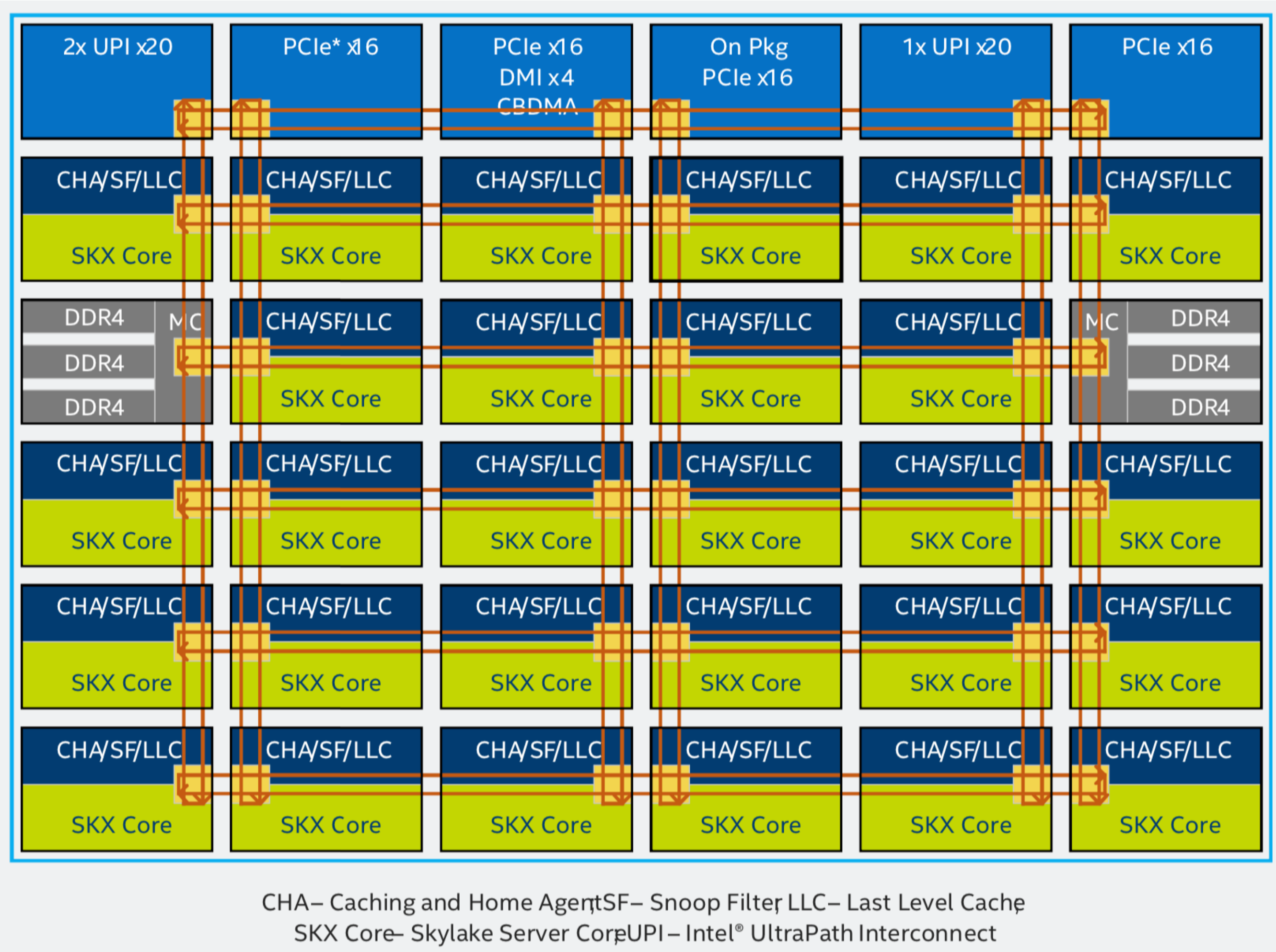

2x UPI x20
PCIe* x16
PCIe x16
DMI x4
CBDMA
On Pkg
PCIe x16
1x UPI x20
PCIe x16
CHA/SF/LLC
SKX Core
DDR4
DDR4
DDR4
MC
MC
DDR4
DDR4
DDR4
CHA– Caching and Home Agent; SF– Snoop Filter; LLC– Last Level Cache;
SKX Core– Skylake Server Core; UPI – Intel® UltraPath Interconnect

Figure 1: Layout of the major elements in the largest die size of the Xeon Scalable Processor Family.

The abstraction of the xAPIC numbering from physical layout of the underlying hardware means that we need to define at least one more numbering scheme and corresponding unambiguous nomenclature. For this report, the physical location of a unit on the mesh will be either shown explicitly on a 2D mesh layout analogous to Figure 1 or will use a two-dimensional subscript notation indicating the row and column starting from the upper left of the mesh as illustrated in Table 1.

$\mathrm{IO}_{0,0}$	$\mathrm{IO}_{0,1}$	$\mathrm{IO}_{0,2}$	$\mathrm{IO}_{0,3}$	$\mathrm{IO}_{0,4}$	$\mathrm{IO}_{0,5}$
$\mathrm{T}_{1,0}$	$\mathrm{T}_{1,1}$	$\mathrm{T}_{1,2}$	$\mathrm{T}_{1,3}$	$\mathrm{T}_{1,4}$	$\mathrm{T}_{1,5}$
$\mathrm{IMC}_{2,0}$	$\mathrm{T}_{2,1}$	$\mathrm{T}_{2,2}$	$\mathrm{T}_{2,3}$	$\mathrm{T}_{2,4}$	$\mathrm{IMC}_{2,5}$
$\mathrm{T}_{3,0}$	$\mathrm{T}_{3,1}$	$\mathrm{T}_{3,2}$	$\mathrm{T}_{3,3}$	$\mathrm{T}_{3,4}$	$\mathrm{T}_{3,5}$
$\mathrm{T}_{4,0}$	$\mathrm{T}_{4,1}$	$\mathrm{T}_{4,2}$	$\mathrm{T}_{4,3}$	$\mathrm{T}_{4,4}$	$\mathrm{T}_{4,5}$
$\mathrm{T}_{5,0}$	$\mathrm{T}_{5,1}$	$\mathrm{T}_{5,2}$	$\mathrm{T}_{5,3}$	$\mathrm{T}_{5,4}$	$\mathrm{T}_{5,5}$

Table 1: Indexing by (row, column) for IO blocks, Memory Controllers (IMC), and Core+CHA Tiles (T), with the same spatial layout as Figure 1.

Within a “tile”, there is a “core” with its private L1 and L2 caches, and a “CHA/SF/LLC” block. Although these units are co-located, there is no “affinity” between the core and the CHA/SF/LLC block. Data accesses that *hit* in the core's private L1 and L2 caches remain local. Accesses that *miss* in a core's private L1 and L2 caches are distributed across *all* CHA/SF/LLC blocks in the chip using an undocumented pseudo-random hash function. (This allows any core to access all CHA/SF/LLC resources on the chip, while distributing accesses approximately uniformly across those resources.) For L2 miss events, the interface from the core to the mesh inspects the physical address being accessed, computes the hash function, and sends a directed message on the mesh to the CHA/SF/LLC responsible for that address.[1]

Upon arriving at the target CHA/SF/LLC, the units perform their specific functions:

- The “Caching and Home Agent” (CHA) is responsible for the generation and ordering of the coherence transactions for each request.
- The “Snoop Filter” (SF) maintains a sparse, inclusive directory of the private caches in the chip, indicating which (if any) additional private caches need to be involved in responding to a request.
- The “Last Level Cache” (LLC, or L3) holds lines recently evicted from L2 caches.

In the interest of brevity, the terms “CHA” or “L3 slice” will be considered synonymous with “CHA/SF/LLC”.

Table 1 defines an unambiguous numbering scheme for the physical positions on the chip – what user-visible numbering will these be mapped to?

- User-visible numbering for the cores is the “logical processor” number.
    - These are explicitly controllable using standard Linux operating system interfaces such as `taskset`, `numactl`, `sched_[gs]etaffinity(),` or binding options with OpenMP runtime libraries.
    - The relation between logical processor number, socket, and “core id” is described in [2].
- User-visible numbering for the CHA/SF/LLC slices is implicit in the interface to the performance counters in those units.
    - The CHA/SF/LLC performance monitoring interface is MSR-based, with a block of 12 contiguous MSR numbers assigned to each CHA.
    - The MSR names include reference to CHAs numbered 0 to 27 [3].

3 Resources for Measurement

Information that may be useful in determining the physical location of each of the elements on the chip is primarily available via the “uncore performance counters'”, described in [3]. The CHA, M2M (Mesh-2-Memory

[1] For remote memory, the access request is sent across the mesh to one of the two “Ultra-Path Interface” (UPI) units in the box labelled “2x UPI x20” in the upper left corner of Figure 1, where the request is transmitted to the “home” chip of the physical address for processing. The UPI interface near the upper right side of the figure is not active in the TACC Frontera node configurations.

controller interface), and M3UPI (Mesh to UPI interface) units all support performance counter events related to traffic passing through the Common Mesh Stop (CMS) at that location. For most of the results presented here, the four programmable performance counters in each of the CHA blocks were programmed as described in Table 2. The names of the directions are in double quotes because of complexities that will be discussed in Section 5.

Counter	Event Name	UMask	Description
0	HORZ_RING_BL_IN_USE	LEFT_EVEN + LEFT_ODD	Data entering mesh stop going “LEFT”
1	HORZ_RING_BL_IN_USE	RIGHT_EVEN + RIGHT+ODD	Data entering mesh stop going “RIGHT”
2	VERT_RING_BL_IN_USE	UP_EVEN + UP_ODD	Data entering mesh stop going “UP”
3	VERT_RING_BL_IN_USE	DOWN_EVEN + DOWN_ODD	Data entering mesh stop going “DOWN”

Table 2: Performance monitoring events used for monitoring data traffic (BL ring) on the mesh.

The mesh routing policy is to traverse the Y (up/down) axis until reaching the target row, then traverse the X (left/right) axis to the target mesh stop [4].

An important feature of these counters is they measure traffic only for data *entering* the mesh stop, not for data *exiting* the mesh stop. It is *usually* possible to measuring exiting traffic at the entrance to the next mesh stop, but that data is unavailable if the next mesh stop contains a disabled CHA/SF/LLC, or if the next mesh stop is a different unit type that does not support the mesh traffic performance counters. Note also that the traffic is only counted upon *entering* the mesh stop even when the traffic is transitioning from the Y-axis to the X-axis.

Attempts to use the mesh data traffic performance counter events at the M2M units (co-located with the memory controllers) were unsuccessful, however DRAM CAS Reads could be read at each of the six DRAM channels in each socket. Other uncore counters were not required to determine the locations of the cores and CHAs but were used in related analyses.

4 Methodology

Given the resources mentioned in the previous section, one might expect it to be straightforward to determine core and CHA locations. For example, one might read a large block of memory from a single core, record the mesh data traffic that occurred, and simply look at the results to see the path that the data must have taken from the two memory controllers to the requesting core. Unfortunately, the result of such measurements will be a collection of data that (for most cores) cannot be mapped onto this mesh given the known routing algorithm. A more extensive set of measurements is therefore required to determine which of the assumptions underlying the analysis proved to be incorrect.

The methodology for determining core and CHA positions was initially developed and applied to Xeon Platinum 8160 (“Skylake Xeon”) processors in the Stampede2 system. The Xeon Platinum 8160 is a 24-core processor with 24 enabled CHA/SF/LLC slices. The task of determining the location of the cores and CHAs was made dramatically more difficult in that initial study because the locations of the disabled cores and CHAs were not known and had to be derived as part of an integrated analysis. With the Xeon Platinum 8280 processors in the Frontera system, all cores and CHAs are enabled, so unexpected features in the data collected are due to features of the system, not due to idiosyncrasies of the specific pattern of disabled cores and CHAs on the chip. To keep the descriptions as clear as possible, the remainder of this section and Section 5 will consider only the fully configured 28-core processors, while Section 7 will discuss the extensions required to determine the locations of disabled cores and CHAs.

A potential complication in interpreting the results arises from the symmetry of the mesh layout. This was disambiguated by running additional experiments accessing memory in another socket. Such traffic must be routed via the UPI links in the upper left corner of the mesh.[2] When reading socket 0 IMC 0 from any core in socket 1, the only active mesh link in socket 0 is the "up" link at CHA 0. We therefore conclude that CHA 0 is at location $T_{1,0}$ and IMC 0 is at location $IMC_{2,0}$.

To unambiguously determine the locations of the cores and CHAs, two microbenchmarks were constructed. The first, `CLX_mapper_auto`, implements the algorithm presented in pseudo-code in Figure 2. This code collects mesh traffic data at every CHA before and after a core reads 2 GiB of data from memory. This is repeated with the process bound to each of the cores in the socket. After the data is collected, a post-processing step compares the delta counts for each core, CHA, and any link showing ≥ 8/9 of the expected traffic is marked "active".[3] This test is reading data from memory controllers on both sides of the package. Given the simple Y-X routing used, it is easy to see that *only the CHA that is co-located with a core will have two active mesh inputs* (one for data coming from IMC0 and one for data coming from IMC1), while the remaining CHAs will have either one active input (if they are on the route from the memory controller to the core), or zero active inputs. For each core, the code reports which CHA (if any) has exactly 2 active mesh links. An error message is reported if no such CHA is found for a core, or if more than one CHA has two active links – indicating interference from other processes running on the system. In practice, the code returns exactly one co-located CHA for each core in almost every run on a non-shared compute node. (All of the raw data is reported as well, for both visual review and in case additional post-processing is required.)

[2] The third UPI link (at $IO_{0,4}$) is not enabled in any of TACC's 2-socket systems.

[3] The specific cutoff depends on the amount of background activity in the system. The value needs to be a little bit less than 1 because there will typically be a small amount of data retained in the L3 cache that will (in general) arrive at the requesting core via a different route than the data from memory.

```
Algorithm 1 Algorithmic structure for CLX_mapper_auto
    allocate and initialize 2GiB array in the memory of socket 0
    program CHA and IMC performance counters to measure data traffic on mesh

    {—Phase 1: run read test on each core and collect CHA and IMC performance counter data —}
    for core =0 to 27 do
        bind to core in socket 0 (i.e., logical processor number 2 x core)
        read all CHA and IMC performance counters ⇒ $counts_{before}$
        read array (sum reduction)
        read all CHA and IMC performance counters ⇒ $counts_{after}$
    end for

    {— Phase 2: post-processing —}
    for core =0 to 27 do
        for CHA=0 to 27 do
            for counter = 0 to 3 do
                $delta \Leftarrow counts_{after} - counts_{before}$
                if $delta \geq 8/9 * expected$ then
                    mark link active
                end if
            end for
            if 2 links are active then
                    report that core and CHA are co-located
            end if
        end for
    end for
```

Figure 2: Outline of the algorithmic structure for CLX_mapper_auto.

The second code, `CLX_mapper_imc0`, implements a very similar algorithm, except that instead of loading a contiguous array, the program loads only cache lines from the three DRAM channels of memory controller 0. This version requires attention to a number of implementation details that are worth noting.

First, unless Sub-NUMA-Clustering (SNC) mode is enabled, addresses within each 4KiB page are distributed across both memory controllers. Even if the addresses accessed are all managed by IMC0, *the core under test must have its hardware prefetchers disabled* to prevent high levels of unintended traffic from IMC1.[4]

Second, the mapping of physical addresses to memory controllers is not fully described in public documents, nor are the corresponding configuration registers visible to users, so additional pre-processing is required. Starting at the beginning of the data array every 8th index (one element per cache line) is tested, and a vector of indices mapping to IMC0 is generated. The test simply loads an address 1000 times (flushing the address after each load) between reads of the memory controller performance counters.[5] If the number of "active" DRAM channels is exactly one, and that channel belongs to IMC0 in the target socket, then the array index is appended to a vector of indices that map to IMC0. This is repeated until the desired number of indices are found.

[4] The hardware prefetchers can be disabled by writing the value 0xf to MSR 0x1a4 on each Logical Processor (and are re-enabled by writing 0x0 to the same MSRs). https://software.intel.com/en-us/articles/disclosure-of-hw-prefetcher-control-on-some-intel-processors

[5] A relatively low number of iterations was possible because the memory controller performance counters are read directly from user-space using 32-bit unsigned load instructions (after mmap'ing the base of PCI configuration space to a user pointer). Reading the counters via a kernel interface would have been at least an order of magnitude slower and would have generated significant unwanted/confusing memory traffic.

The number of cache lines mapping to IMC0 must be chosen so that the array of indices can fit into the L2 data cache and not generate additional mesh traffic when used. For the 1MiB private L2 caches on Xeon Scalable processors, the maximum number of 32-bit indices that can be retained is 262,144, but in practice a much smaller number is used to keep cache misses on the index vector to a minimum – sizes in the range of 64Ki elements to 128Ki elements work well. After performing a number of tests, it became clear that on the Frontera compute nodes, even-numbered 256-Byte blocks are assigned to IMC0, while odd-numbered blocks are assigned to IMC1. A variant of the code was produced to exploit this feature, avoiding the need to use an index vector.

Once the vector of indices mapping to IMC0 has been produced, the code performs the same measurements as `CLX_mapper_auto`, binding to each core in turn and measuring the data traffic on the entire mesh.

The summary output of `CLX_mapper_imc0` is a list of the active links in the system, by package, CHA, and CHA counter number, along with a count of the total number of active links in each package. As with `CLX_mapper_auto` the raw counts are reported as well, for sanity-checking and additional post-processing. IMC counters are wrapped around the tests for each core to confirm proper placement of the data on IMC0, with typical runs showing about 99.8% of all memory traffic coming from IMC0.

5 Expectations and Initial Observations

Without knowing where the numbered units are located, it is not possible to jump directly to a visualization of the data. Instead, for each physical core, the number of "active" links in the "up", "down", "left", and "right" directions were counted and compared to expected values. Figure 3 provides an example of the *expected* mesh data traffic for DRAM reads made by the core in the upper left corner of the processor ($T_{1,0}$). Recall that the mesh counters are on the *input* to each mesh stop. Only the CHA at $T_{1,0}$ is expected to show active mesh links on two sides.

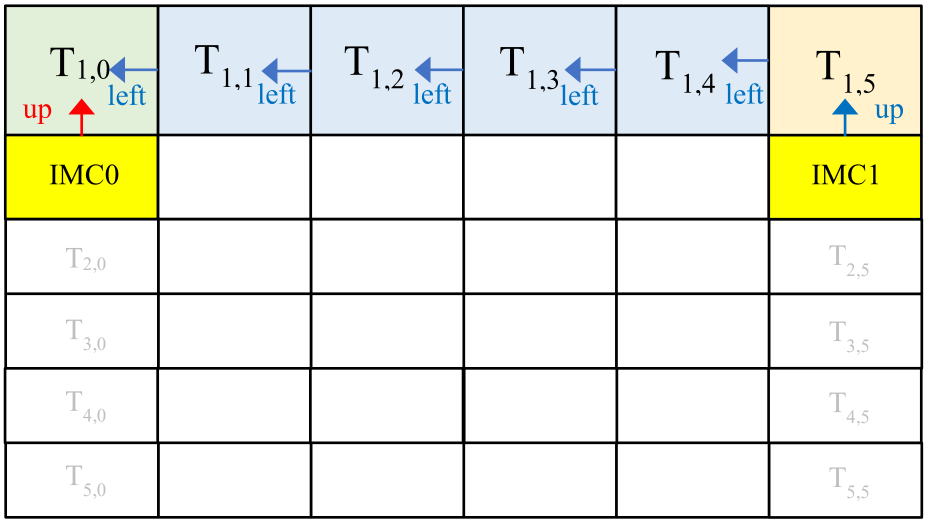


Figure 3: Expected "active" mesh links (at the input to the mesh stop) when the core at $T_{1,0}$ (light green) reads from both memory controllers. Red arrow and text indicate traffic from IMC0, blue arrow and text indicate traffic from IMC1. The CHA at $T_{1,5}$ (light orange) receives traffic from IMC1 in the "up" direction and transfers it to the horizontal mesh. The four CHAs in light blue receive left-bound traffic and pass it on to the next mesh stop to the left until arriving at the requesting tile.

Following the example in Figure 3 for each core, we expect to see some clear patterns in the *number* of active links, even if the link *locations* are not initially known.

Property Set 1: Grouping cores by row
The routing of data traffic on the mesh should give the cores in each row unique properties:

- First row: 6 cores will have 2 CHAs with active links in the "up" direction.
    - These will be $T_{1,0}$ and $T_{1,5}$, immediately above the memory controllers.
- Second row: 4 cores will have no CHAs with active links in either the "up" or "down" directions.
- Third row: 6 cores will have 2 CHAs with active links in the "down" direction.
    - These will be $T_{3,0}$ and $T_{3,5}$, immediately below the memory controllers.
- Fourth row: 6 cores will have 4 CHAs with active links in the "down" direction.
    - Including the 2 CHAs from the 3rd row plus $T_{4,0}$ and $T_{4,5}$.
- Fifth row: 6 cores will have 6 CHAs with active links in the "down" direction.
    - Including the CHAs from the 3rd and 4th rows, plus $T_{5,0}$ and $T_{5,5}$.

Observations: Measurements on the 28-core Xeon Platinum 8280 processors matched these properties for the "up" and "down" links in all systems tested. (Processors with less than 28 enabled cores and CHAs will be considered in Section 7.)

Property Set 2: Core-to-CHA Mapping
As discussed in Section 4, we expect that for each core the measurements will include exactly one CHA with active (inbound) links on two sides, and that across all cores this will provide a 1:1 mapping of cores to CHAs.
Observations:

- This 1:1 mapping was observed in the measurements on all Xeon Scalable Processors tested.
- For processors with the same number of enabled cores and CHAs (e.g., 28-core/28-CHA on the Xeon Platinum 8280, 24-core/24-CHA on the Xeon Platinum 8160), the mapping was found to be *identical* across nodes when the nodes were the same server model from the same vendor with the same processor model installed.
- The core-to-CHA mappings were *not* identical across server models with different distributions of Logical Processor numbers across sockets (i.e., interleaved vs block-distributed, as discussed in [2]), even with identical processors and from the same vendor.
- The core-to-CHA mappings were *not* identical across nodes on a small number of test nodes with 24-core/26-CHA Xeon Platinum 8260 processors – though these were early access systems and may not have the same properties as production-level systems.

Property Set 3: Grouping cores by column
A third expected pattern comes from considering the cores by column:

- First column: 4 cores with have 5 CHAs with mesh traffic going in the "left" direction.
    - This includes the co-located CHA and the four other CHAs in the same row – excluding the right-hand column.
- Second column: 5 cores
    - Traffic to the "right" at the co-located CHA.
    - Traffic to the "left" at the three other CHAs in the same row (excluding the right-hand column).
- Third column: 5 cores
    - Traffic to the "right" at the co-located CHA.
    - Traffic to the "right" at the CHA in the same row in the second column.
    - Traffic to the "left" in the CHAs of the two columns to the right.
- The last columns are expected to repeat with mirror symmetry.
- Therefore, within one of the full rows of six cores, the expected ["left","right"] counts by column are:
    - [5,0], [4,1], [3,2], [2,3], [1,4], [0,5]
- For the row with the IMCs, the expected counts are [4,1], [3,2], [2,3], [1,4] (excluding cores at each end).

Observations: *None of the measurements on any systems matched these patterns.* For each core, the sum of the active "left" and "right" links was 5 (as expected), but the split into "left" and "right" did not follow the expected pattern. In fact, every core generated the same counts: 3 active "left" links and 2 active "right" links. These links were activated in the CHAs on the same row as the requesting core, as expected, but otherwise did not make immediate sense.

6 Layout of Fully Configured 28-core/28-CHA Processors

Many attempts to re-imagine the mesh numbering were attempted, as were interpretations of the mesh as a set of "flattened rings" (alternating single-hop and double-hop connections to minimize the length of the longest connection (e.g., [2]), but none of these explained the observations.

Eventually, a "Eureka" occurred, and it was realized that if one assumes that the meanings of "left" and "right" are swapped in alternating columns, the results exactly match the expected patterns. Compensating for the left/right mirroring by columns causes the counts of locations of active links to precisely match expectations, and the methodology of Section 5 is then sufficient to fully determine the physical locations of each of the numbered core and CHA units (and to allow consistent visualizations of up/down/left/right mesh data traffic).

For the Frontera two-socket compute nodes, the layout of the cores and CHAs is presented in Figure 4. This pattern applies to all 8008 of the original Frontera two-socket compute nodes but may not apply to the same processors in different server models.

$IO_{0,0}$	$IO_{0,1}$	$IO_{0,2}$	$IO_{0,3}$	$IO_{0,4}$	$IO_{0,5}$
0	4	36	26	50	2
IMC0	32	24	54	6	IMC1
28	20	52	10	34	30
16	48	12	38	18	14
44	8	40	22	46	42

Logical Processor locations

$IO_{0,0}$	$IO_{0,1}$	$IO_{0,2}$	$IO_{0,3}$	$IO_{0,4}$	$IO_{0,5}$
0	4	9	14	19	24
IMC0	5	10	15	20	IMC1
1	6	11	16	21	25
2	7	12	17	22	26
3	8	13	18	23	27

CHA/SF/LLC locations

Figure 4: (LEFT) Locations of Logical Processors in socket 0 of Frontera Xeon Platinum 8280 compute nodes. In socket 1, the pattern is the same, but the Logical Processor numbers are larger by 1. (RIGHT) Locations of the numbered CHAs in Frontera compute nodes. The CHA numbering is local, so this pattern applies directly to both sockets.

It is immediately evident that the CHA numbers follow a simple pattern, starting with zero in the upper left corner, incrementing down the column (skipping over the IMC block), shifting a row to the right, and repeating until CHA 27 is located in the lower right corner. This pattern of CHA numbering was seen on all systems with Xeon Scalable Processors (with modifications for disabled CHAs, as discussed below in Section 7).

It is likewise evident that the Logical Processor numbers follow a much more subtle pattern. (Note that these are all even numbers because the socket-alternating Logical Processor numbering scheme in the Frontera compute nodes places all the odd-numbered Logical Processors in socket 1 – where the pattern is exactly analogous). Unlike the pattern of CHA numbering, the pattern of Logical Processor numbering is different for nodes with socket-interleaved Logical Processor distribution and nodes with block-distributed Logical Processor numbering.

Examination of the die photo of the 28-core Xeon Scalable Processor shows that the layout of the processor cores is left-right reversed in alternating columns, as seen in Figure 5. Without special logic to detect and adjust for this mirroring, it is not surprising that this could cause the observed reversal of the meanings of "left" and "right" in alternating columns.

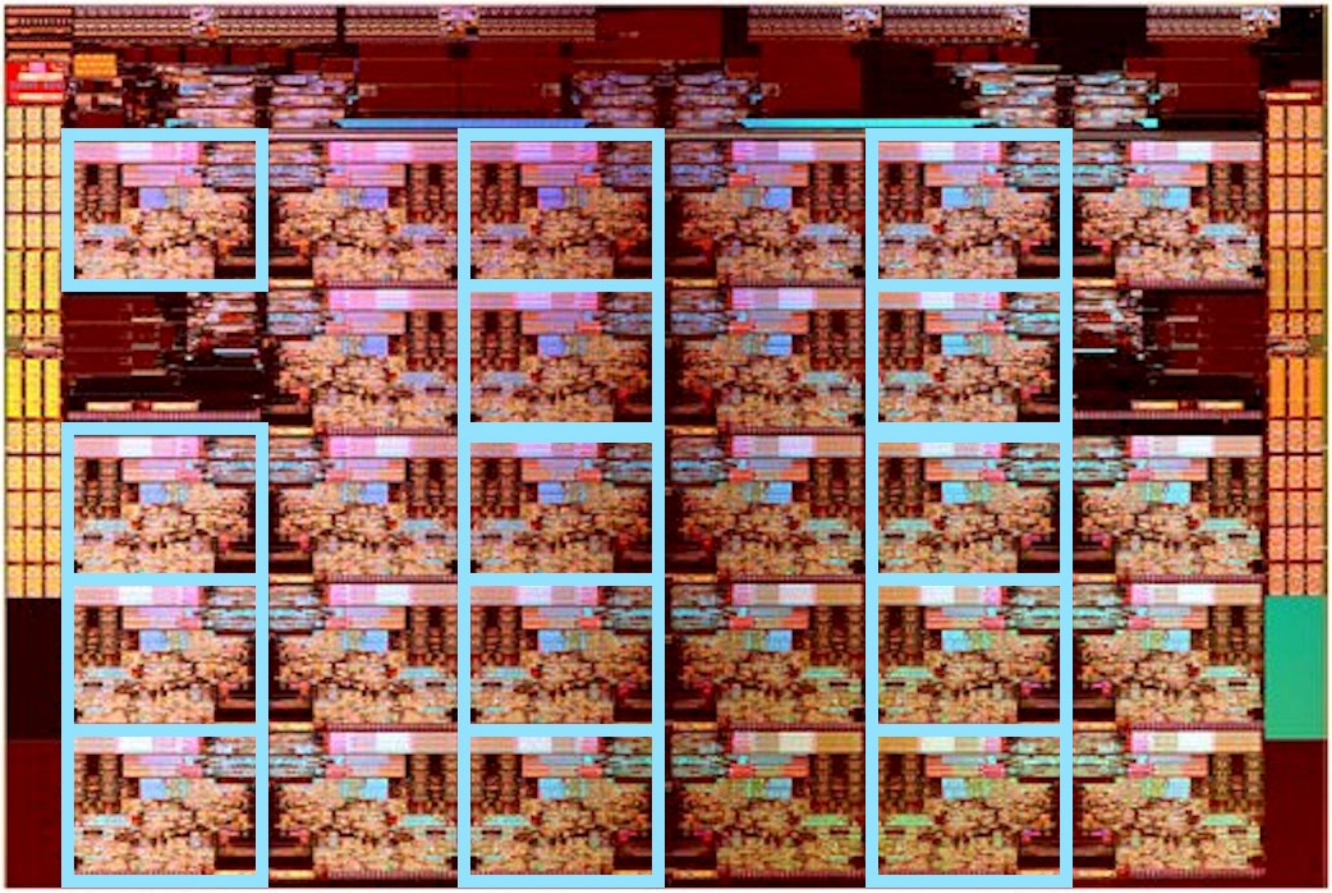


Figure 5: Die photo of the XCC (28-core) Xeon Scalable Processor. The cores in columns 0, 2, and 4 (outlined in light blue) are in the "standard" orientation, while the cores in columns 1, 3, and 5 are laid out with left-to-right mirroring. (Image from [4], annotations added.)

7 Analysis with Disabled Tiles

For the 24-core/24-CHA Xeon Platinum 8160 processors used in the initial study, the difficulty of the analysis was compounded by knowing neither the overall mesh layout nor the locations of the disabled tiles. It was also not certain at the time how the mesh routing worked with disabled tiles, or whether the performance counters were functional in CHAs with disabled LLCs.

Perhaps the first important result was the recognition that the 24 CHAs associated with "active" LLC slices are accessed via the MSRs for CHAs 0 to 23, with the remaining four CHAs "inactive" – i.e., the performance counters were not responsive to writes to the MSRs for CHAs 24 to 27 (though the MSR accesses were not blocked). This means that there is an indirection layer in the hardware between the CHA number used by the MSRs and the physical CHAs on the chip, so CHA numbers cannot, by themselves, directly indicate location.

The patterns of active links described in Section 5 were initially envisioned in the slightly more general context of a chip with four disabled cores and four disabled tiles – all in unknown locations. It was quickly recognized that the 1:1 mapping of cores to CHAs ("Property Set 2") holds on all of these processors. Since every core has a co-located CHA, it must also be true that the disabled cores and disabled CHAs are co-located – significantly simplifying the possible topologies that must be considered.[6]

It was also clear in the early analyses that the distribution of "up" and "down" traffic ("Property Set 1") was consistent with expected results for chips with four disabled CHAs. Due to the Y-first routing, traffic coming from the memory controllers only moves vertically in the first and last columns. A disabled CHA in either of those columns will result in a missing "vertical" link count for all of the cores that are on that row or on a row further away from the memory controller in the same direction, as illustrated in Figure 6 for a hypothetical chip with disabled CHAs at $T_{4,0}$ and $T_{1,5}$. These observed distributions of the vertical traffic were also sufficient to confirm that data traffic travels *through* mesh stops co-located with disabled CHAs, rather than requiring a more complex routing algorithm. I.e., the actual links traversed do not change, only the number of CHAs that count the traffic changes.

$IO_{0,0}$	$IO_{0,1}$	$IO_{0,2}$	$IO_{0,3}$	$IO_{0,4}$	$IO_{0,5}$
1 up	*1 up*	*1 up*	*1 up*	*1 up*	*disabled*
IMC0	0 up/down	0 up/down	0 up/down	0 up/down	IMC1
2 down	2 down	2 down	2 down	2 down	2 down
disabled	3 down	3 down	3 down	3 down	3 down
5 down	5 down	5 down	5 down	5 down	5 down

Vertical traffic for this row with no disabled CHAs
2 up
0 up/down
2 down
4 down
6 down

Figure 6: Impact of disabled CHAs in the IMC columns on vertical mesh traffic counts generated by cores reading from both memory controllers. The value in each tile indicates the total number of CHAs whose performance counters show vertical traffic counted when the core at that location reads from both memory controllers. Note that all of that vertical traffic takes place in columns 0 and 5 (marked with heavy outlines). Values in blue italics are those modified by the disabled CHA above IMC1 ($T_{1,5}$). Values in red are those modified by the disabled CHA below IMC0 ($T_{4,0}$).

Attempting to find the appropriate extensions to mesh traffic counts when grouping cores by column ("Property Set 3") was extremely frustrating until the reversal of the meaning of "left" and "right" in alternating columns was discovered. Once the left-right mirroring is understood, it becomes relatively easy to review the results manually and apply the observed link traffic counts to unambiguously map cores and CHAs to locations on the mesh. The number of special cases to consider leaves the procedure rather difficult to automate, so a search for a better understanding of the algorithm(s) used to number the cores and CHAs in the presence of disabled tiles was initiated.

Xeon Scalable Processors have a hardware register called CAPID6 in PCI Configuration Space that contains a 28-bit bitmap of the enabled LLC slices (documented in Section 1.7.1 of [3]). The documentation refers only to the number of bits set in this register, not to the bit locations, but the fact that it is a bit map (rather than simply a count of active CHAs) suggests investigating whether the bit locations may relate to fixed physical locations on the die. The bus numbers for the PCI configuration space registers may vary from system to system, but on the 2-socket Xeon Scalable Processor systems at TACC, the values can be obtained by a system administrator using the shell commands:

```
    # setpci -s 17:1e.3 0x9c.l
    # setpci -s 85:1e.3 0x9c.l
```

[6] Many Xeon Scalable Processor models have more enabled LLC slices than cores, but TACC does not have any of these models in production systems. Limited testing in early access systems with 24-core/26-CHA Xeon Platinum 8260 processors suggests that these have slightly more complex properties than the 24-core/24-CHA processors discussed in this section.

On the Stampede2 test node c591-101, the first command returns the CAPID6 value from socket 0 and the second returns the CAPID6 value from socket 1:

```
    0f7dfbef
    0fef77bf
```

The CAPID6 values were collected for almost all (94%) of the Xeon Platinum 8160 processors in the Stampede2 system (including both the Dell nodes in the production cluster and a number of Intel S2600 nodes in test clusters), and 120 unique CAPID6 bitmap patterns were found. All of the bitmap patterns have exactly two bits cleared in the upper 14 bits and exactly two bits cleared in the lower 14 bits, but otherwise show no obvious symmetries or patterns. A handful of nodes with different CAPID6 values were chosen for manual analysis and comparison between disabled CHA locations based on mesh traffic measurements and cleared bit fields in CAPID6.

Reviewing the results showed that the CHAs are numbered using a simple extension of the numbering for the fully configured chips. The CHAs are numbered consecutively starting at the upper left, going down, and then right, but skipping over the disabled CHAs as well as skipping over the IMCs.

$IO_{0,0}$	$IO_{0,1}$	$IO_{0,2}$	$IO_{0,3}$	$IO_{0,4}$	$IO_{0,5}$
0	4	9	14	19	24
IMC0	5	10	15	20	IMC1
1	6	11	16	21	25
2	7	12	17	22	26
3	8	13	18	23	27

CHA numbering in 28-CHA chip

$IO_{0,0}$	$IO_{0,1}$	$IO_{0,2}$	$IO_{0,3}$	$IO_{0,4}$	$IO_{0,5}$
0	disabled	8	12	16	20
IMC0	4	disabled	15	17	IMC1
1	5	9	14	18	21
2	6	10	disabled	19	22
3	7	11	15	disabled	23

CHA numbering with 4 disabled CHAs

Figure 7: (Right) CHA numbering in a Xeon Platinum 8160 processor with CAPID6 0x0f7dfbef. Cleared bits 4, 10, 17, 23 in CAPID6 correspond to disabled CHAs at the positions of the default numbered CHAs 4, 10, 17, 23 (left).

Attempts to derive an algorithm for the core numbering based on physical position were unsuccessful. However, the mapping of core number to co-located CHA number is the same in all of the nodes of the same vendor server model (and same processor), so it suffices to use CAPID6 register to determine the locations of the active CHA numbers, and then use the measured core-to-CHA mapping to map the locations of the numbered cores.

The CAPID6 value of `0x0f7dfbef` was the most commonly observed pattern in the system, accounting for approximately 23% of all processors. This results in the same CHA numbering in both Dell and Intel systems, but different mapping of core numbers to locations, as shown in Figure 8. If a processor with the same CAPID6 value were installed in socket 1, the corresponding logical processor numbers in the Dell system are obtained by adding 1 to the values in Figure 8 (left), and the corresponding logical processor numbers in the Intel system are obtained by adding 24 to the values in Figure 8 (right). There are tantalizing glimpses of comprehensible patterns in the layout of the core numbers in both systems, but changes to CAPID6 result in completely different-looking patterns, making a table look-up a simple and effective choice.

$IO_{0,0}$	$IO_{0,1}$	$IO_{0,2}$	$IO_{0,3}$	$IO_{0,4}$	$IO_{0,5}$
0	disabled	8	10	6	2
IMC0	4	disabled	34	30	IMC1
24	28	32	22	18	26
12	16	20	disabled	42	14
36	40	44	46	disabled	38

Core Numbering in Dell C6420 node

$IO_{0,0}$	$IO_{0,1}$	$IO_{0,2}$	$IO_{0,3}$	$IO_{0,4}$	$IO_{0,5}$
0	disabled	2	3	4	5
IMC0	1	disabled	15	16	IMC1
12	13	14	9	10	17
6	7	8	disabled	22	11
18	19	20	21	disabled	23

Core Numbering in Intel S2600 node

Figure 8: Socket 0 locations of numbered Logical Processors in two systems equipped with Xeon Platinum 8160 processors and CAPID6 values of 0x0f7dfbef. The Dell C6420 node alternates logical processor numbers between sockets (even/odd, so all the logical processor numbers shown here for socket 0 are even), while the Intel 2600 uses a block distribution (first half/second half, putting logical processors 24 to 47 in socket 1).

As discussed in [2], the CHA numbering scheme results in the first half of the CHA numbers being located on the left side of the chip and the second half of the numbers on the right side.

When "Sub-NUMA Clustering" (SNC) mode is enabled, the CHAs, cores, and memory controllers on the left side of the chip are assigned to one NUMA node and those on the right side of the chip are assigned to another NUMA node. The physical processor cores assigned to each NUMA node are the same, but the numbering of both the cores and the NUMA nodes themselves differs between the two numbering schemes.

- In the socket-interleaved case (Figure 8, left), the assignments are:
  - Logical processors [0,4,8,12,16,20,24,28,32,36,40,44] in the left NUMA node.
  - Logical processors [2,6,10,14,18,22,26,30,34,38,42,46] in the right NUMA node.
  - In socket 0, the left NUMA node is node 0 and the right NUMA node is node 2.
  - In socket 1, the left NUMA node is node 1 and the right NUMA node is node 3.
- In the block-distributed numbering case (Figure 8, right), the assignments are:
  - Logical processors [0,1,2,6,7,8,12,13,14,18,19,20] are in the left NUMA node.
  - Logical processors [3,4,5,9,10,11,15,16,17,21,22,23] are in the right NUMA node.
  - In socket 0, the left NUMA node is node 0 and the right NUMA node is node 1.
  - In socket 1, the left NUMA node is node 2 and the right NUMA node is node 3.

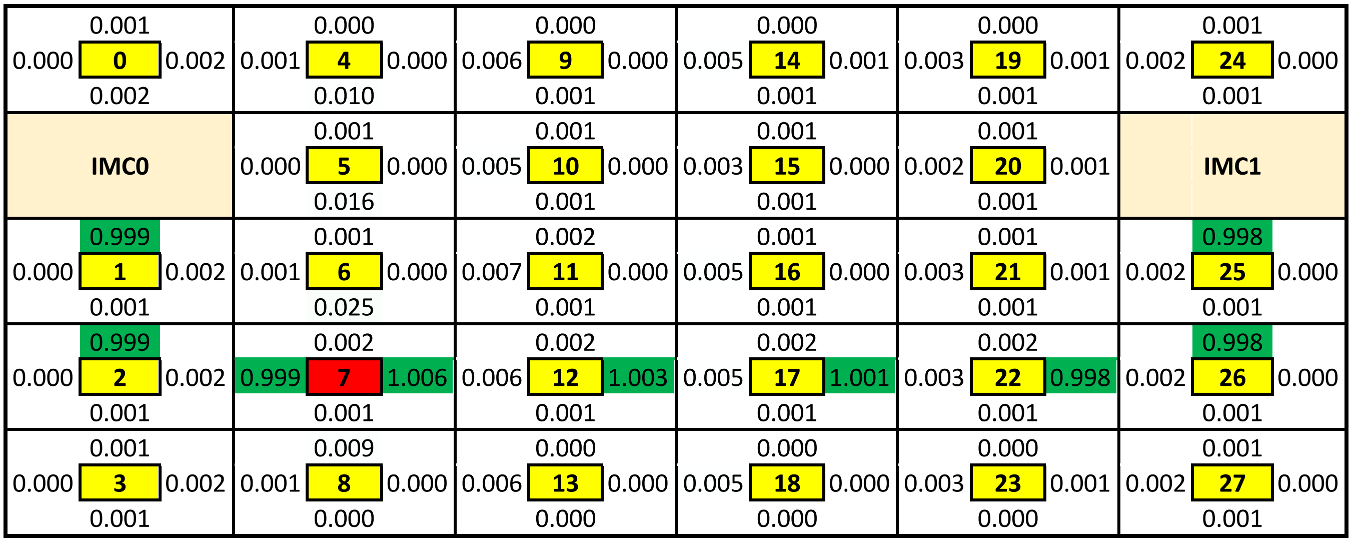
Looking for patterns, we note that in the interleaved scheme, bit 2 of the logical processor ID is the same as the high-order bit of the NUMA node number, while the package number is the low-order NUMA node number. In the block-distributed scheme, it appears that converting a logical processor ID to the low-order bit of the NUMA node number requires a division operation (or a table lookup). It is unclear whether these patterns have any intrinsic relationship with the capabilities of the hardware, or if they are essentially "accidents of history".

Summary of important results ***for the Xeon Platinum 8160***:

- CHAs associated with "inactive" LLC slices are "inactive", but the corresponding mesh stop is functional – passing data through and switching from Y to X directions as needed.
  - Performance counters are not available at the mesh stop for inactive CHAs.
- The MSR interface to the CHA/SF/LLC units maps the "P" active units to indices 0 to "P-1".
  - CHA numbers are therefore not directly related to physical location.
  - The MSR interfaces for CHAs "P" to 27 remain writable, but the performance counters return only zero values.
- All active cores are co-located with active CHAs – likewise for inactive cores and CHAs.
- The mapping of Logical Processor number to co-located CHA number is identical in instances of the same server model with the Xeon Platinum 8160 processor but can differ across models and vendors.

8 An Example Application

Using the understanding of the core and CHA numbering (along with the inversions of the horizontal mesh directions), the output of `CLX_mapper_auto` and `CLX_mapper_imc0` can be mapped onto a schematic of the processor chip to allow visual examination of traffic patterns. Figure 9 shows the normalized mesh traffic in socket 0 of a Frontera compute node while a thread bound to Logical Processor 48 (co-located with CHA 7) is reading from both memory controllers. Figure 10 shows the traffic when reading data from IMC0 only, using the index-based version of `CLX_mapper_imc0` and a list of 131,072 cache lines mapping to IMC0. In each of these figures, the label in the center of each box is the CHA number, and the values adjacent to the four edges are the quantity of incoming mesh data traffic entering from the adjacent box, normalized by the expected traffic for an active link.



0.001 0.000 **0** 0.002 0.002	0.000 0.001 **4** 0.000 0.010	0.000 0.006 **9** 0.000 0.001	0.000 0.005 **14** 0.001 0.001	0.000 0.003 **19** 0.001 0.001	0.001 0.002 **24** 0.000 0.001
IMC0	0.001 0.000 **5** 0.000 0.016	0.001 0.005 **10** 0.000 0.001	0.001 0.003 **15** 0.000 0.001	0.001 0.002 **20** 0.001 0.001	**IMC1**
0.999 0.000 **1** 0.002 0.001	0.001 0.001 **6** 0.000 0.025	0.002 0.007 **11** 0.000 0.001	0.001 0.005 **16** 0.000 0.001	0.001 0.003 **21** 0.001 0.001	0.998 0.002 **25** 0.000 0.001
0.999 0.000 **2** 0.002 0.001	0.002 0.999 **7** 1.006 0.001	0.002 0.006 **12** 1.003 0.001	0.002 0.005 **17** 1.001 0.001	0.002 0.003 **22** 0.998 0.001	0.998 0.002 **26** 0.000 0.001
0.001 0.000 **3** 0.002 0.001	0.009 0.001 **8** 0.000 0.000	0.000 0.006 **13** 0.000 0.000	0.000 0.005 **18** 0.000 0.000	0.000 0.003 **23** 0.001 0.000	0.001 0.002 **27** 0.000 0.001

Figure 9: Normalized mesh traffic measurements from `CLX_mapper_auto` (reading from both memory controllers) on socket 0 of a Frontera compute node. Mesh locations are labeled by CHA number (in yellow). Values at the edges of each box indicate the inbound mesh data traffic coming from the adjacent box. The core under test is Logical Processor 48, co-located with CHA 7 (highlighted in red).

In Figure 9, the data is quite clean, with three active links carrying the traffic from IMC0 to the requesting core and six active links carrying the traffic from IMC1 to the requesting core. These nine active links show a range of 0.998 to 1.006 times the expected traffic of 32Mi increments per active link[7]. The most active link off of these expected paths is the inbound traffic in the UP direction at CHA 6 (immediately above the tile with the requesting core) at only 2.5% of the nominal active link traffic. (The data for socket 1 is not shown but has a maximum normalized value of less than 0.0005 for this case.)

More detailed inspection of the patterns of the extra traffic near Core 48/CHA 7 shows a reasonable correlation with the pattern expected if we assume that a small fraction of clean L2 victims are being transferred from Core 48's L2 cache to the distributed L3 cache. Looking at the inbound links of the four adjacent tiles (CHAs 2,6,8,12), the aggregate (normalized) outbound traffic from the tile containing Core 48/CHA 7 is 0.0424. If this corresponds to clean L2 victim traffic, it is reasonable to expect it to be uniformly distributed across the 28 L3 slices, and the Y-X routing scheme allows a simple computation of the fraction of the traffic that is expected to be sent in each direction. Table 3 shows a near perfect match between the observed distribution of outbound traffic and the expected distribution for L2 victim lines being uniformly distributed across the L3 slices on the chip. The

[7] The 2GiB array has 32Mi cache lines, 16Mi of which will come from each IMC. The mesh interface is ½ cache line in width, so each cache line transfer will increment the corresponding mesh data traffic counter twice.

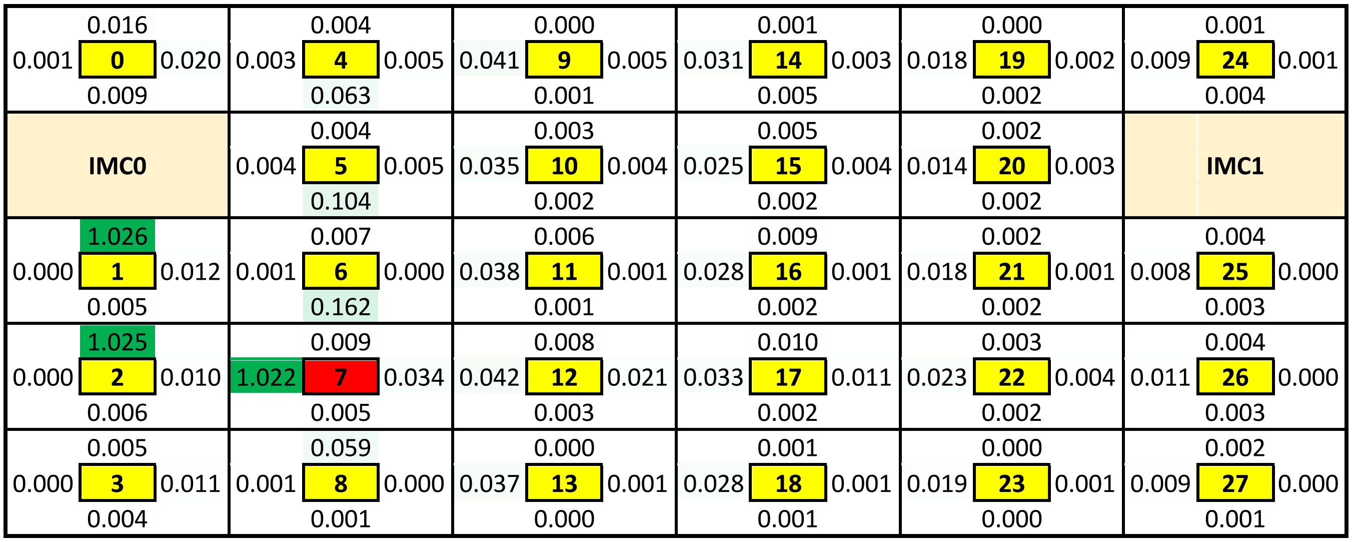
fraction of L2 victims being sent to the L3 is small – about 2.1% of the total. For the remainder, the hardware predicts that the lines would be unlikely to be reused from the L3, so they are dropped.[8]

Direction	Number of LLCs routed via this direction	Expected fraction of outbound traffic in this direction	Observed fraction of outbound traffic in this direction
UP	16	59.3%	58.4%
DOWN	6	22.2%	22.1%
LEFT	1	3.7%	4.2%
RIGHT	4	14.8%	15.2%

Table 3: Distribution of outbound data traffic from the tile containing Core 48/CHA 7. "Expected" traffic distribution is based on the assumption that outbound data traffic is being distributed uniformly across the L3 slices of the chip.

Figure 10 shows the results for reading `CLX_mapper_imc0` for the test running on the same logical processor (48, co-located with CHA 7). Here the number of cache lines loaded is limited by the number of indices that can be held in the L2 cache, so the expected traffic per active link is only 1/128th of the expected traffic for the case shown in Figure 9. The three active links carrying data from IMC0 are the same as in Figure 9, but it is apparent that the "noise" is significantly larger in this "one-sided" case.

The core performance counter IDI_MISC.WB_UPGRADE reports that only 2.6% of L2 victims are being sent to the L3, while the sum of the outbound counts from the core under test is about 27% of the array volume loaded. The difference is accounted for by the overhead required to read the uncore performance counters. Because these are accessed via MSRs they can only be read in kernel mode. The Linux */dev/cpu/*/msr* device driver only allows one MSR read per kernel call, so the code requires 224 kernel calls to read the four performance counters for each CHA in each socket. By removing the code under test and repeating the experiment, it was confirmed that most (65% - 90%, depending on methodology) of the excess mesh data traffic was due to kernel calls to read the uncore performance counters.



0.016 0.001 **0** 0.020 0.009	0.004 0.003 **4** 0.005 0.063	0.000 0.041 **9** 0.005 0.001	0.001 0.031 **14** 0.003 0.005	0.000 0.018 **19** 0.002 0.002	0.001 0.009 **24** 0.001 0.004
IMC0	0.004 0.004 **5** 0.005 0.104	0.003 0.035 **10** 0.004 0.002	0.005 0.025 **15** 0.004 0.002	0.002 0.014 **20** 0.003 0.002	**IMC1**
1.026 0.000 **1** 0.012 0.005	0.007 0.001 **6** 0.000 0.162	0.006 0.038 **11** 0.001 0.001	0.009 0.028 **16** 0.001 0.002	0.002 0.018 **21** 0.001 0.002	0.004 0.008 **25** 0.000 0.003
1.025 0.000 **2** 0.010 0.006	0.009 1.022 **7** 0.034 0.005	0.008 0.042 **12** 0.021 0.003	0.010 0.033 **17** 0.011 0.002	0.003 0.023 **22** 0.004 0.002	0.004 0.011 **26** 0.000 0.003
0.005 0.000 **3** 0.011 0.004	0.059 0.001 **8** 0.000 0.001	0.000 0.037 **13** 0.001 0.000	0.001 0.028 **18** 0.001 0.001	0.000 0.019 **23** 0.001 0.000	0.002 0.009 **27** 0.000 0.001

Figure 10: Normalized mesh traffic measurements from `CLX_mapper_imc0` (reading from only IMC0) on socket 0 of a Frontera compute node. Mesh locations are labeled by CHA number (in yellow). Values at the edges of each box indicate the inbound mesh data traffic coming from the adjacent box. The core under test is Logical Processor 48, co-located with CHA 7 (highlighted in red). The maximum normalized data traffic on any link in socket 1 is 0.013.

[8] Since the test is read-only, almost all lines in the L2 cache are clean.

9 Summary

Although it requires some degree of system-specific validation, the methodology described here makes it possible to unambiguously map from user-visible core and CHA numbers to locations on the die in Xeon Scalable Processors. In addition, this methodology has uncovered an unexpected transformation in the meanings of "LEFT" and "RIGHT" in the on-chip mesh that has almost certainly frustrated prior efforts to use these mesh traffic counters.

The combination of the core/CHA mapping and clarification of mesh data traffic directions should enable a large number of new research studies related to the performance characterization of these processors – particularly those related to mesh throughput limitations and to latency optimization of cache-to-cache transfers.

Caveat Emptor:

The mappings reported here (Figure 4 for the standard compute nodes in the Frontera system, Figure 7 and Figure 8 for the Stampede2 SKX partitions) have been verified on subsets of both systems, and there is no reason to expect that any of the standard compute nodes will be configured differently. There is, however, *ample* reason to be cautious about applying these results to other systems – even those with the same processor model from the same vendor. The different mapping schemes described in this report and in [2] may not be the only options available.

10 Acknowledgments

This work has been supported by grants from the National Science Foundation, including award numbers 1663578 and 1854828.

11 References

[1] A. Kumar, "The New Intel Xeon Scalable Processor (Formerly Skylake-SP)," presented at the Hot Chips 29, Aug. 2017, [Online]. Available: https://www.hotchips.org/wp-content/uploads/hc_archives/hc29/HC29.22-Tuesday-Pub/HC29.22.90-Server-Pub/HC29.22.930-Xeon-Skylake-sp-Kumar-Intel.pdf.

[2] McCalpin, John D., "Observations on Core Numbering and 'Core ID's' in Intel Processors," Texas Advanced Computing Center, Technical Report TR-2020-01, Nov. 2020. [Online]. Available: http://dx.doi.org/10.26153/tsw/10858.

[3] Intel, "Intel Xeon Processor Scalable Memory Family Uncore Performance Monitoring Reference Manual." Intel Corporation, Jul. 2017, [Online]. Available: https://software.intel.com/content/dam/develop/public/us/en/documents/336274-intel-xeon-processor-scalable-memory-family-uncore.pdf.

[4] S. M. Tam *et al.*, "SkyLake-SP: A 14nm 28-Core Xeon processor," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 34–36, doi: 10.1109/ISSCC.2018.8310170.