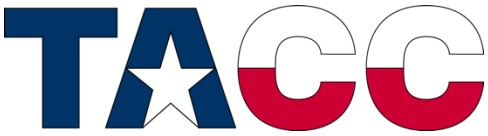

ACELab Technical Report TR-2020-01

Observations on Core Numbering and "Core ID's" in Intel Processors

Document Revision 2.0
November 30, 2020
Status: Release, Active

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

1. Introduction

While Intel processors have many mysterious, undocumented features, one of the first that users become aware of is that there are (at least) two ways that logical processor numbers are distributed among sockets in multi-socket servers. Many systems alternate logical processor numbers across sockets, while others use a block distribution. A few systems use more idiosyncratic schemes. It does not appear to be possible to predict the distribution scheme based on the processor model or system vendor or vendor server model, nor does it appear to be possible to change the distribution scheme used by a particular system.

Looking slightly deeper, several interfaces in the Linux operating system provide access to a "core id" field. This is clearly *related* to the logical processor numbering, but with most processor models the "core id" field contains gaps in the otherwise contiguous numbering (while logical processor numbers are always contiguous, starting at zero[1]). In most cases, the gaps occur at exactly the same values for any processor with the same number of enabled cores, but not in all cases. In most cases, the mapping of logical processor numbers to "core id" is monotonic, but not in all cases.

Taken together, these observations raise a number of questions, such as:

- Are the "core id" patterns related to the location of the cores on the die?
- Are the "core id" patterns related to disabled cores on the die?
- Are there required relationships between logical processor numbers and "core id" numbers?

In response to such questions, this report describes and analyzes the patterns of logical processor distribution and "core id" values in recent and current TACC systems. The goal is two-fold:

- To provide users, system administrators, and curious hobbyists with a high-level understanding of what these values mean (and don't mean), so that they can be (mostly) ignored hereafter.
- To provide analysts and computer architects with background material for future reports that will describe the full mapping of logical processor numbers to cores at specific locations on the processor die for several different processor models.

Sections 3 and 4 of this report review and summarize the patterns used for logical processor ("LProc") distribution on Intel-processor-based systems at TACC, showing that most systems use one of two simple numbering schemes. The exceptions are documented as a reminder that the most common patterns are not obligatory.

Sections 5, 6, and 7 focus on the "core id" patterns on Intel-processor-based systems at TACC. In many cases the pattern is the same for every processor of the same core count in a cluster, while in other cases the pattern varies on a package-by-package basis. Comparing patterns across many different processor models and generations provides some insights into the properties of the underlying mappings.

Section 8 discusses the mapping of logical processor numbers to "core id" numbers, showing two "traditional" patterns and an intriguing new pattern in systems using Xeon Scalable Processors ("Skylake Xeon" and "Cascade Lake Xeon").

[1] The operating system is allowed to disable cores (leaving gaps in the list of active cores), but there is always an underlying 1:1 mapping of the N execution contexts in a system to the contiguous logical processor numbers [0…N-1].

2. Background: Clarifying the Nomenclature

Our nomenclature for computer systems developed when computers were much simpler and is now responsible for considerable confusion. The words "processor" and "core" are especially ambiguous.

For this report:

- *Cluster* – a collection of servers that are collectively administered and scheduled.
- *Node (or System)* – a single server.
- *Package (or Socket)* – the physical "thing" that fits in a processor socket.
- *Physical core* – a set of transistors in the package that operate as an independent processing element.
- *Logical processor* – an addressable execution context in a physical core.

In the introduction, the term "distribution" (rather than "mapping") was used as a reminder that the assignment of logical processor numbers to sockets is only one part of the full mapping of logical processor numbers to *package*, *physical core*, and (when applicable) *thread context*.

Processor numbering (and core ids) are limited to a single node[2], so when the operating system (using information provided by the hardware and firmware) sets up the numbering of the logical processors, the contiguous values must be distributed across the three "indices" of *package*, *physical core*, and execution context (or *thread*).

For these three indices, we can enumerate six "regular" distributions – one from each of the six orderings of the three indices from most-rapidly-varying to least-rapidly-varying. E.g., *[package, core, thread]* distributes logical processor numbers across *packages* first, then across *cores*, and finally across *thread* contexts. Most systems use *[package, core, thread]* or *[core, package, thread*], (described in more detail below), but there are a few exceptions – as well as some systems with irregular distributions.

3. Common Patterns of Logical Processor Numbering

Most TACC systems use one of two patterns for the distribution of logical processor numbers to packages and physical cores. These will be referred to as "interleaved" (*[package, core, thread]*) and "block-distributed" (*[core, package, thread]*), as exemplified in Table 1. Note that using the "thread" index as least-rapidly-varying means that the basic numbering for the packages and cores is unchanged when HyperThreading is enabled.

Block-distributed Logical Processor numbering (with or without HyperThreading)

	Package 0		Package 1	
	Thread 0	*Thread 1*	Thread 0	*Thread 1*
Core	0	*24*	12	*36*
Core	1	*25*	13	*37*
Core	2	*26*	14	*38*
Core	3	*27*	15	*39*
…	…	…	…	…
…	…	…	…	…
Core	10	*34*	22	*46*
Core	11	*35*	23	*47*

Socket-interleaved Logical Processor Numbering (with or without HyperThreading)

	Package 0		Package 1	
	Thread 0	*Thread 1*	Thread 0	*Thread 1*
Core	0	*24*	1	*25*
Core	2	*26*	3	*27*
Core	4	*28*	5	*29*
Core	6	*30*	7	*31*
…	…	…	…	…
…	…	…	…	…
Core	20	*44*	21	*45*
Core	22	*46*	23	*47*

Table 1: The two most common logical processor numbering schemes, illustrated for two packages, twelve cores per package, and 1 or 2 threads per core.

[2] The hardware is capable of coordinating the assignment of core ids across nodes, but this feature is not used at TACC.

For single-package systems (e.g., Xeon Phi), there is no "socket-interleaved" option, but if HyperThreading is enabled, there are still two choices: *[thread, core]* and *[core, thread]*. Similarly, with HyperThreading disabled in a multi-package system, there are also two choices: *[package, core]* and *[core, package*].

Table 2 lists the major Intel-based supercomputing clusters at TACC, along with the numbering scheme and an indication of whether the core id pattern is uniform across nodes.

System (queue)	Processor	System Vendor(s)	Total nodes	Logical Processor Numbering	Core id pattern
Frontera (normal)	Xeon Platinum 8280	Dell	8008	[pkg, core]	uniform
Stampede2 (skx-*)	Xeon Platinum 8160	Dell	1736	[pkg, core, thread]	uniform
Stampede2 (!skx-*)	Xeon Phi 7250	Dell, Intel	4200	[core, thread]	irregular
Lonestar5	Xeon E5-2690 v3	Cray	1252	[core, pkg, thread]	uniform
Hikari	Xeon E5-2690 v3	HPE	432	[core, pkg, thread] †	uniform
"test1"	*Xeon Platinum 8160*	*Intel*	*29*	*[core, pkg, thread]*	*uniform*
"test3"	*Xeon Platinum 8260*	*Intel*	*8*	*[core, pkg]*	*irregular*
Cowboy	*Xeon E5-2660 v4*	*Dell*	*11*	*[pkg, core]*	*uniform*
Stampede login	Xeon E5-2680	Dell	4	[pkg, core]	uniform
Stampede	Xeon E5-2680	Dell	6400	[core, pkg]	uniform
Stampede	Xeon Phi SE10P	Intel	6880	<see text>	uniform

Table 2: Characteristics of the TACC systems reviewed in this study. The first five entries are for production systems, followed by test systems (italics), then decommissioned systems (grey text). The entry marked with † has an irregular pattern when HyperThreading is disabled – see Section 4 for discussion.

Note that on the (retired) Stampede system, the login nodes and compute nodes used different logical processor numbering, despite using the same processors and coming from the same vendor. Similarly, the Xeon Platinum 8160 systems in the Stampede2 "skx" queue are numbered differently than nodes with the same processors in the Stampede2 "test1" system (though these systems are from different vendors).

4. Unusual Patterns of Logical Processor Numbering

While almost all systems fit into the two most common patterns, there have been a few unusual patterns in TACC systems at various points in time. Noting these serves as a reminder that the combination of hardware, firmware, and software is not obligated to produce mappings following the most common patterns.

The first unusual pattern has been seen only on the Hikari (HPE) system when the nodes are booted with HyperThreading disabled. Rather than simply using the left (Thread 0) columns in the Block-distributed numbering of Table 1, the pattern in Table 3 is observed, splitting each package into two blocks and alternating blocks across packages. An analogous pattern was seen for service nodes in the same system equipped with 10-core Xeon E5-2660 v4) processors, i.e., logical processors [0…4,10…14] in socket 0 and logical processors [5…9,15…19] in socket 1.

Half-Block-distributed Logical Processor numbering
(only seen with HyperThreading disabled)

	Package 0		Package 1	
	Thread 0	Thread 1	Thread 0	Thread 1
Core	0		6	
Core	1		7	
Core	2		8	
Core	3		9	
Core	4		10	
Core	5		11	
Core	12		18	
Core	13		19	
Core	14		20	
Core	15		21	
Core	16		22	
Core	17		23	

Table 3: Unusual logical processor numbering seen in the Hikari system with HyperThreading disabled.

The second unusual pattern was seen only with the first-generation Xeon Phi SE10P (a.k.a., "Knights Corner", KNC, "mic") processors (in the retired Stampede system). In these 61-core processors, a basic *[thread, core]* distribution was modified by shifting logical processor 0 to core number 60 – presumably to shift poorly-distributed operating system code off of core 0 and onto the highest-numbered core in the system. Logical processor numbering then continued as in Table 4.

Xeon Phi x100 ("KNC") Logical Processor Numbering

	Package 0			
	Thread 0	Thread 1	Thread 2	Thread 3
Core	1	2	3	4
Core	5	6	7	8
Core	9	10	11	12
Core	13	14	15	16
Core	…	…	…	…
Core	…	…	…	…
Core	237	238	239	240
Core	0	241	242	243

Table 4: Logical Processor numbering of Xeon Phi x100 ("Knights Corner") moves logical processor 0 to the highest-numbered core.

5. About the "core id"

The "core id" is a unique identifier provided by the hardware/firmware for each logical processor. These identifiers are the "targets" associated with the "Advanced Programmable Interrupt Controller" (APIC) functionality of the hardware, so the operating system must track these in order to properly target interrupts. The current Intel processors at TACC all support the third-generation "x2APIC" architecture, though the hardware operates in second-generation "xAPIC" mode on all processors except the Xeon Phi x200 nodes.

In "xAPIC" mode, the hardware and firmware produce a unique 8-bit "xAPIC ID" for each logical processor in a node. This mode supports addressing of 256 logical processors, which is sufficient for all TACC systems except those with Xeon Phi x200 processors (68 physical cores * 4 threads/core = 272 logical processors). The 8-bit "xAPIC ID" packs together bit fields that indicate the package, physical core, and thread context for each logical

processor. In "x2APIC" mode, the hardware and firmware produce a unique 32-bit "x2APIC ID" for each logical processor in a node. In either mode, a logical processor can execute the CPUID instruction to retrieve both its xAPIC/x2APIC ID and information about how many bits are used to indicate the package, the physical core, and the thread context. For the Xeon Platinum 8280 processors and Xeon Platinum 8160 processors, the 8-bit ID is constructed (and deconstructed) as shown in Table 5, along with a proposed nomenclature to minimize ambiguity when referring to these values.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
(global) xAPIC ID							
Package ID		"local" xAPIC LProc ID					
(global) xAPIC core ID							Thread ID
Package ID		"local" xAPIC core ID					Thread ID

Table 5: Bit fields and naming conventions of the xAPIC/x2APIC ID for 2-socket Xeon Scalable Processors with more than 16 physical cores. Note that the "thread" bit is reserved even if HyperThreading is disabled in the BIOS.

The architectural definition [1] [2] of the xAPIC/x2APIC ID only requires that the values be unique for each logical processor. The observations clearly show that contiguous numbering is not required, but it is unclear whether there are other required properties.

In Linux systems, the value of "core id" in `/sys/devices/system/cpu/cpu*/topology/core_id` and `/proc/cpuinfo` is the "local xAPIC core ID" from Table 5. This value is also used by the Intel OpenMP runtime to report the mapping of logical processors to package/core/thread when the KMP_AFFINITY environment variable includes the "verbose" clause. Since these are commonly used interfaces, many users have observed (and been confused by) the different patterns of values displayed.

6. Common "uniform" core id patterns

For (almost) all processors except the Xeon Phi x100 and x200, the "local xAPIC core ID" numbering depends on the number of physical cores in each package as shown in Table 6.

Physical Cores per Package	"local" xAPIC/x2APIC core numbers used
8	0 1 2 3 4 5 6 7
10	0 1 2 3 4 8 9 10 11 12
12	0 1 2 3 4 5 8 9 10 11 12 13
14	0 1 2 3 4 5 6 8 9 10 11 12 13 14
16	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
18	0 1 2 3 4 8 9 10 11 16 17 18 19 20 24 25 26 27
20	0 1 2 3 4 8 9 10 11 12 16 17 18 19 20 24 25 26 27 28
24	0 1 2 3 4 5 8 9 10 11 12 13 16 17 18 19 20 21 24 25 26 27 28 29
28	0 1 2 3 4 5 6 8 9 10 11 12 13 14 16 17 18 19 20 21 22 24 25 26 27 28 29 30

Table 6: "Local xAPIC ID" numbers used as a function of physical cores per package. Applies to Xeon E5 v1/v2/v3/v4, and Xeon Scalable Processors.

These core ID patterns hold for systems from any of our vendors (Cray, Dell, HPE, Intel – except for the Stampede2 "test3" nodes that will be discussed in Section 7), and for processor generations from "Sandy Bridge EP" through "Cascade Lake Xeon". The patterns are identical on systems with interleaved or block-distributed logical processor numbering. The patterns are not affected by the presence or absence of HyperThreading (except

that the values are repeated for the second thread context in each core with the "thread ID" bit set to 1), or by enabling "Sub-NUMA-Cluster" (or "Cluster-on-Die") mode[3].

Note that the core ID sequences contain "skips" even on processors with all cores active (such as the 28-core Xeon Platinum 8280). For the Xeon Scalable Processors, the CAPID6 register provides a 28-bit map of the enabled/disabled L3 slices [3]. Review of this register in more than 400 24-core processors of the Stampede2 "skx" partitions showed that identical xAPIC ID sequences were used across 55 different patterns of disabled cores. Core ID sequences were also seen to be independent of disabled cores in Xeon E5 v3 ("Haswell EP") and Xeon E5 v4 ("Broadwell EP") systems.

For these processor counts, the numbering scheme is clearly dividing the core numbers into "clumps" of up to 8 contiguous core numbers starting on multiple-of-8 boundaries. These "clumps" are of uniform size for most core counts – i.e., every value in the table except for 18 (discussed below).

One way to interpret this numbering scheme is as a recursive bisection to create groups of no more than 8 cores. E.g., for 24 physical cores per package, splitting the cores in half gives two groups of 12. The first group is numbered starting at zero and the second group is numbered starting at the next power of 2 larger than 12: 16. The groups are larger than 8, so they are subdivided again into contiguous groups of 6 values that start on multiple-of-8 boundaries. If a group has an odd number of elements, it is split as evenly as possible, with the larger "half" assigned to the lower-numbered group and the smaller "half" assigned to the higher-numbered group. This matches all the observed cases and predicts that the 22-core and 26-core processors (not currently available for testing) would split their local xAPIC core IDs into contiguous blocks of length [6,5,6,5] and [7,6,7,6], respectively.

Comparing the use of the bit fields in the xAPIC ID (Table 5) with the observed patterns (Table 6) suggests that support for "Sub-NUMA-Cluster"/"Cluster-on-Die" mode may be the reason for some properties of the mapping. Splitting the xAPIC IDs evenly between a 0-based lower half and a power-of-2-based upper half allows the high-order bit of the "local" xAPIC ID to be treated as a "package" bit for the specification of the "within-package" NUMA node.

Whether this is required or not, it was observed in all "mainstream" Xeon processors that the lower half of the local xAPIC core IDs were located in the left half of the die and that the upper half of the local xAPIC core IDs were located in the right half of the die. The details will be presented in later reports.

7. Irregular core id patterns

Except for the Xeon Phi processors, *almost* all of the "mainstream" Xeon processors have consistent xAPIC ID patterns. The only exceptions have been in a small number of pre-production systems with second-generation Xeon Scalable Processors. A set of four Intel S2600 two-socket servers with pre-production 24-core Xeon Platinum 8260L processors show three different patterns of xAPIC IDs – none of which match the pattern seen on the 24-core Xeon Platinum 8160 systems. There is no apparent correlation with the CAPID6 fields, but the sample size is too small to detect anything but very simple relationships. Another two S2600 servers equipped with pre-production 24-core Xeon Platinum 8260Y processors show the same pattern across all four sockets, matching the pattern seen on three of the eight nodes with 8260L processors. It is not clear whether there is any *meaning* to these differences, but it does serve to demonstrate that the hardware does not require the most commonly observed patterns.

[3] "Cluster-on-Die" mode configures the package to operate as two separate NUMA nodes. The half of the memory channels and the half of the physical cores on the left side of the chip form one NUMA node, and the memory channels and cores on the right side of the chip form a second NUMA node.

The Xeon Phi x100 ("Knights Corner") Model SE10P processors in the (retired) Stampede1 system at TACC used linear numbering of the 61 (of 62) enabled cores. Each core supported four logical processors, resulting in 244 logical processors – numbered 0 to 243. Preliminary testing suggested that the numbering was linear with position on the ring (skipping over the disabled core), but the testing was not comprehensive. Note that the SE10P processor was a "Special Edition" model that was also marked as an "Engineering Sample".

In contrast to the Xeon Phi x100 Model SE10P, the Xeon Phi 7250 ("Knights Landing") processors in the TACC Stampede2 "KNL" partition have a direct mapping between location on the die and the x2APIC ID. The die has 38 "tiles", each supporting two cores. For the model 7250, 68 cores are enabled, with 4 tiles having both cores disabled. Reviewing the x2APIC ID patterns on 458 of the 504 Stampede2 KNL nodes from Intel, we saw that the disabled cores were associated with the tiles immediately above and below the two DRAM memory controllers on approximately 2/3 of processors, with the remaining disabled tiles spread more-or-less uniformly around the rest of the chip. A total of 221 unique patterns of skipped x2APIC IDs were seen on this set of nodes, with 40% seen only once. Similar patterns, but with slightly different statistics, were seen on a sample including 757 of the 3696 Stampede2 KNL nodes from Dell.

Because the x2APIC IDs correspond directly to locations on the die, this topic will be investigated in more detail in a separate report.

8. Mapping of Logical Processor numbers to "core id" values

For the processors with "uniform" sets of xAPIC IDs, it was noted above that the sets of values appearing in the list(s) of xAPIC IDs are not trivially related to the distribution of logical processor numbers across sockets. However, the *mapping* of logical processor numbers to xAPIC IDs does show only a few patterns, and these are related to the pattern of logical processor distribution.

For processors using block-distributed logical processor numbering (*[core, package, thread]*), the mapping of logical processor numbers observed on these systems is always monotonic. Logical processor zero maps to (local) xAPIC ID 0, logical processor 1 maps to the next value present in the (local) xAPIC ID list, etc.

For processors using interleaved logical processor numbering (*[package, core, thread]*), two patterns were observed: the monotonic pattern seen with block-distributed logical processor numbering, and a completely different (and very interesting) permutation pattern. With the permuted pattern, contiguous (local) logical processors are mapped to a specific class of permutations of the contiguous blocks of local xAPIC IDs.

As an example, consider the 28-core Xeon Platinum 8280 processors in the Frontera compute nodes. These have four groups of 7 contiguous local x2APIC values: [0…6, 8…14, 16…22, 24…30]. For socket 0, the local logical processors are [0, 2, 4, … , 52, 54], so I can define a "local" logical core ID as the logical processor number divided by 2. The mapping of the first 14 local logical core IDs to local xAPIC ids shown in Table 7. The mapping of the other half is analogous, adding 14 to the local logical processor number and 16 to the local xAPIC ID. Socket 1 values show the same pattern (after subtracting 1 from the global logical processor number).

local logical processor	0	1	2	3	4	5	6	7	8	9	10	11	12	13
local xAPIC ID	0	6	1	5	2	4	3	14	8	13	9	12	10	11

Table 7: Mapping of local logical processor ID to local xAPIC ID in 28-core processors with socket-interleaved logical processor numbering. The mapping is analogous for the upper half of the logical processors, with 14 added to the local logical processor number and 16 added to the local xAPIC ID.

The mapping may not represent an obvious pattern, but with a bit of spatial rearrangement it becomes clear that the interleaving matches the pattern seen when "squashing" a set of rings into a flat, interleaved topology, as shown in Figure 1 for the first 14 physical cores. (The pattern for the other 14 is analogous.)

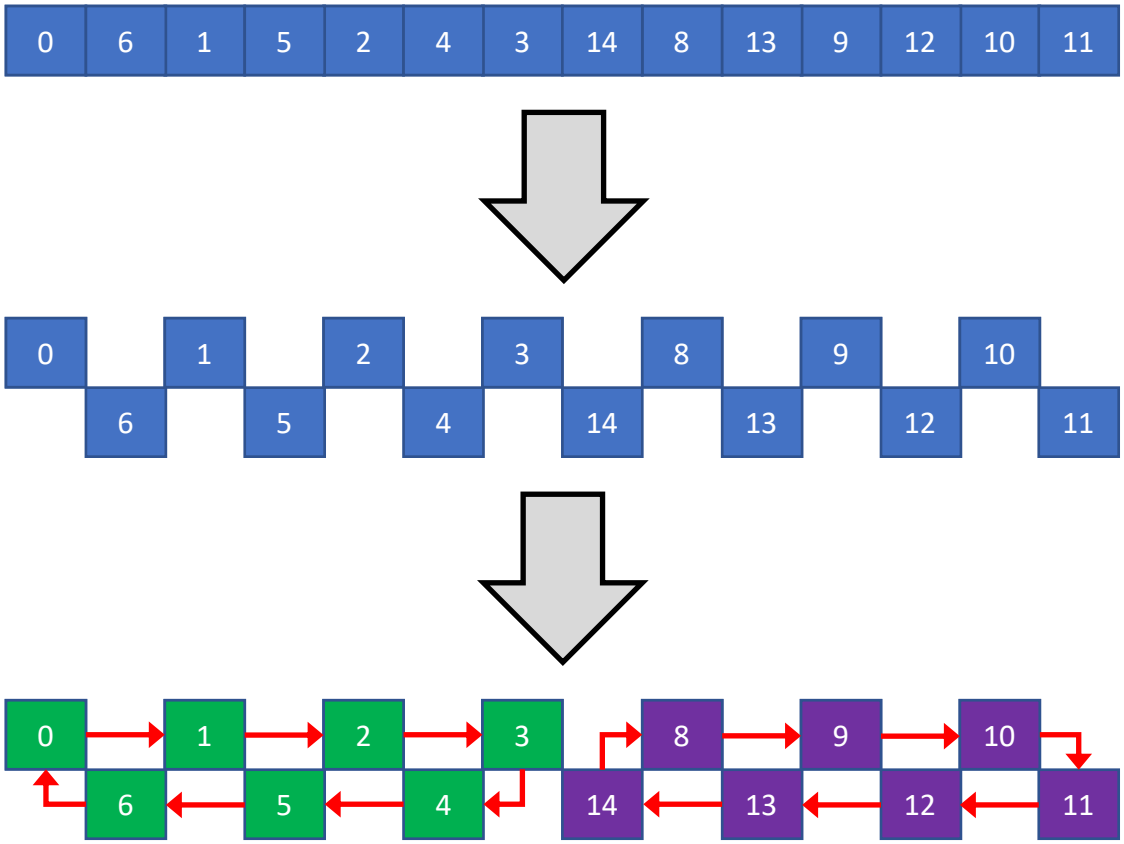

Figure 1: Observed local xAPIC ID permutation matches the "squashing" of rings (each composed of a contiguous group of IDs).

Here there is a change in start of the "ring" layout between the first 7 cores and the second 7 cores. This shift is only seen when there are an odd number of contiguous IDs in each "chunk". For contiguous local xAPIC IDs blocks of *even* size, the permutation structure repeats identically for each of the two or four blocks per package.

This unusual "squashed ring" permutation has been observed on all TACC's Dell systems for Xeon Scalable Processors that use interleaved logical processor numbering (*[package, core]* or *[package, core, thread]*). None of our Intel-processor-based systems from other vendors currently use interleaved logical processor numbering, so it is not clear whether this is a feature of the reference firmware or a vendor-specific idiosyncrasy.

Note that this permutation operator only applies *within* groups of contiguous local xAPIC core IDs, so it is consistent with the observation at the end of Section 6 that the lower half of the values are associated with cores located in the left half of the die and the upper half with cores located in the right half of the die.

9. Summary

This report documents the logical processor numbering, the xAPIC numbering, and the mapping of logical processors to xAPIC IDs in TACC's Intel-processor-based systems.

For multi-socket systems, the logical processor distribution is either block-distributed (*[core, package, thread]*) or interleaved (*[package, core, thread]*). For these systems, the "core id" is unique 8-bit value for each logical processor, with no direct relationship to the location of the core on the die.

For the Xeon Phi x200 systems, the logical processor numbering is core-interleaved (*[core, thread]*), and the "core id" has a 1:1 mapping with the physical location on the processor die, so that gaps in the local x2APIC core id field correspond directly to disabled cores.

This information serves as background to the (far more interesting) discussion of the mapping of logical processor numbers to physical locations on the processor die, which will be the subject of future reports. Once the mapping of logical processors to physical locations is available, many additional studies become possible – such as investigation of traffic on the two-dimensional mesh interconnect, investigation of thermal variations across the die, or investigation of the spatial patterns of disabled cores and/or caches on less-than-fully-configured processors.

10. Acknowledgments

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

11. References

[1] Intel Corporation, "Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 2," 16 November 2020. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html.

[2] Intel Corporation, "Intel 64 and IA-32 Architectures Software Developer's Manual: Volume 3," 16 November 2020. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html.

[3] Intel Corporation, "Intel Xeon Processor Scalable Memory Family Uncore Performance Monitoring Reference Manual," July 2017. [Online]. Available: https://software.intel.com/content/www/us/en/develop/download/intel-xeon-processor-scalable-memory-family-uncore-performance-monitoring-reference-manual.html.