Exploration of Memory and Cluster Modes in Directory-Based Many-Core CMPs

Subodha Charles
University of Florida
Gainesville, FL, USA
charles@cise.ufl.edu

Chetan Arvind Patil
Arizona State University
Tempe, AZ, USA
chetanpatil@asu.edu

Umit Y. Ogras
Arizona State University
Tempe, AZ, USA
umit@asu.edu

Prabhat Mishra
University of Florida
Gainesville, FL, USA
prabhat@ufl.edu

***Abstract*—Networks-on-chip have become the standard interconnect solution to address the communication requirements of many-core chip multiprocessors. It is well-known that network performance and power consumption depend critically on the traffic load. The network traffic itself is a function of not only the application, but also the cache coherence protocol, and memory controller/directory locations. Communication between the distributed directory to memory can introduce hotspots, since the number of memory controllers is much smaller than the number of cores. Therefore, it is critical to account for directory-memory communication, and model them accurately in architecture simulators. This paper analyzes the impact of directory-memory traffic and different memory and cluster modes on the NoC traffic and system performance. We demonstrate that unrealistic models in a widely used multiprocessor simulator produce misleading power and performance predictions. Finally, we evaluate different memory and cluster modes supported by Intel Xeon-Phi processors, and validate our models on four different cache coherence protocols.**

I. Introduction

Continuous advances in manufacturing technologies enable integrating an increasing number of general purpose as well as specialized processors on the same chip. For example, Intel Xeon Phi processors, code-named “Knight’s Landing” (KNL), feature 64-72 Atom cores and 144 vector processing units [1]. A larger number of cores can be exploited only if each core has *fast* and *high bandwidth* access to memory. Therefore, a low-latency network-on-chip (NoC) interconnects the cores with each other and a suite of integrated memory controllers (MC), which provide interfaces to multi-channel DRAM (MCDRAM) and main memory (DDR) [2]–[4].

Modern chip multiprocessor (CMP) architectures commonly employ directory-based cache coherence protocols and multiple levels of cache. The first and second level caches (L1 and L2) are collocated with each core, while the last level cache (LLC) and tag directory are distributed throughout the chip, as illustrated in Figure 1. An L2 miss triggers a request to the directory that keeps track of the corresponding memory address. In case of a hit, the data is returned from (or written to) the LLC slice collocated with the directory. Otherwise, the request is forwarded to one of the MCs. Due to pin limitations and packaging constraints, the number of MCs is much less than the cores. For example, Intel Xeon Phi has 8 MCs interfacing MCDRAM and two MCs interfacing DRAM, while the system has 72 cores [1]. Similarly, AMD Opteron 6386 SE has 16 cores with 1 MC and 4 memory channels. Therefore, LLC-Memory communication exhibits a many-to-few communication pattern while Core-LLC communication is many-to-many. In other words, memory traffic is likely to introduce *hotspots*, whereas Core-LLC traffic is relatively uniform. As a result, these hotspots and poor design choices can cause significant performance degradation [5], as demonstrated in our experimental results.

As an example, Intel Xeon Phi processor provides different cluster modes that define the affinity of directories to specific MCs due to the importance of LLC/directory to memory traffic. In the *all-to-all* mode, any directory can send requests to any MC on the chip. In contrast, *quadrant* mode divides the chip into four virtual partitions, where the directories in each partition are paired with specific controllers in their own quadrant. As a result, the *quadrant* mode localizes the traffic in an attempt to improve the memory performance. Similarly, *sub-NUMA* cluster modes SNC-2 and SNC-4 divide the processor in two and four virtual sockets, respectively. In addition, the MCDRAM can be used as a cache (*Cache mode*),

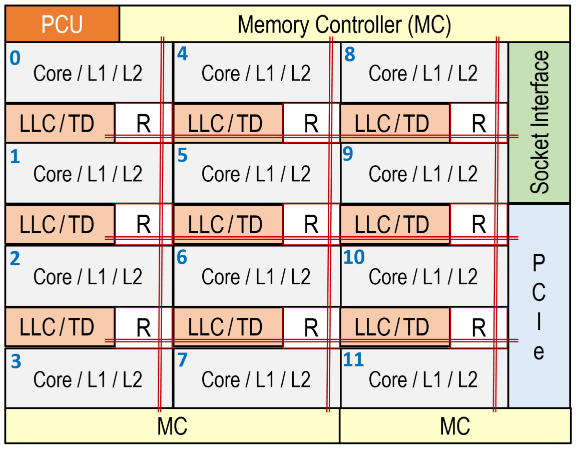


Fig. 1: Representative illustration of a many-core CMP. **L1/L2:** Private first and second level cache, **LLC:** Distributed shared last level cache, **TD:** Tag directory distributed along with LLC, **PCIe:** PCI-express controller for I/O units, **R:** Router, **PCU:** Power control unit.

This work was partially supported by the National Science Foundation (NSF) grants CNS-1526687 and CNS-1526562.

an extension to DDR (*Flat mode*) or in a *Hybrid mode* [1], giving three memory mode options. Each of these choices lead to a different NoC traffic pattern as a function of the workload.

The optimum cluster and memory mode is a strong function of the target application. Applications whose threads and memory footprint fit to a single quadrant can take advantage of the strong locality of *quadrant* and *sub-NUMA* modes. However, highly parallel applications with a large number of threads and memory footprint may benefit from *all-to-all* and *flat* memory modes. Analyzing the power consumption and performance impact of cluster and memory modes is important for two reasons. First, it enables us to use the existing platforms optimally. Second, it can help in making better architectural choices. This analysis is not feasible on existing hardware platforms, since the traffic between the cores and memory is not observable. Furthermore, there are no public simulators capable of performing this exploration. For example, gem5 [6], which is one of the most widely used architecture simulators, assumes that there is an interface from each tile to the main memory. Consequently, the memory access latency is modeled, but the actual traffic from LLC/directory to memory is not captured. This makes the default gem5 model unsuitable for cluster and memory mode exploration.

This paper analyzes the impact of cluster and memory mode choices on the NoC traffic. We demonstrate that congestion on the NoC links affects not only the communication latency, but also power consumption and application execution time. We also show that any exploration that involves LLC/directory to memory traffic requires a simulation framework that models the cache coherence protocols accurately. We demonstrate both qualitatively and quantitatively that neglecting the LLC/memory traffic, as it is done in gem5, gives highly optimistic results in terms of the network load, latency and power consumption. We also show that this inaccuracy can lead to misleading conclusions in terms of optimal MC placement. Then, we describe how the LLC/directory to memory traffic, originating from directory-based cache coherence, can be modeled in architectural simulators. Using the corrected gem5 model, we evaluate the power consumption and performance impact of *quadrant* and *all-to-all* cluster modes together with *cache* and *flat* memory modes which configure the directory-MC affinity. *The major contributions of this paper are as follows:*

- We demonstrate the importance of modeling the directory-memory traffic, and show that considering only core to directory traffic gives highly optimistic results.
- We describe how to accurately model the LLC/directory to memory traffic, and contrast it with the assumption adopted in gem5 [6] full system simulator.
- We explore speed-up achieved by different cluster and memory modes supported by the state-of-the-art CMPs using the enhanced version of gem5. We confirm that the trends are the same with those obtained on the real platform.
- We demonstrate the impact of this work on four different cache coherence protocols.

The rest of the paper is organized as follows. Section II presents related work on NoC design and exploration. Section III gives a background on memory and cluster modes. Section IV presents our NoC modeling and exploration framework. Section V presents the experimental results. Finally, Section VI concludes the paper.

II. Related Work

Prior work on traffic exploration on NoC and optimization motivates the need for better memory and processor placement to reduce contention and latency. Early work in this area suggests the efficient distribution of memory traffic to provide quality-of-service guarantees [7]. Abts et al. [5] tackle the problem of optimum MC placement where m cores need to be placed with n MCs. The placement is decided by examining the variation in latency experienced by cores to access each MC. "Diamond" placement is found to be the best for an 8x8 mesh with 16 MCs, while further improvements are achieved by introducing a class-based deterministic routing algorithm. Xu et al. [8] leverage this idea to find an optimal placement for the same configuration. The minimum number of MCs and their placement required to achieve a given performance goal was explored by taking Intel SCC [9] as a case study [10]. Once the number of MCs are decided and placed, it creates opportunity for optimization by dynamically mapping workload data to appropriate MCs [11].

The effect of modeling the main memory access through the directory was discussed by Duraisamy et al. [12]. They explore the traffic patterns of two-level MESI directory protocol and AMD's Hammer-based HyperTransport (HT) [13] protocol to design an efficient multicast aware wireless NoC. Ros et al. analyzed area and traffic trade-offs associated with cache coherence protocols [14]. To optimize power and performance, Schuchhardt et al. [15] propose a method to place directories closer to their shared data and thereby eliminating many network traversals. Other coherence traffic-based optimization techniques include coherence protocol deactivation for private block accesses to reduce directory accesses [16], and a bloom filter mechanism for tagless coherence directory [17].

In contrast to our work, none of the prior studies rigorously explore the affinity between PE, MC and directory in a system running directory-based cache coherence and optimization with different cluster and memory modes. As shown in our experimental results, the conclusions of the optimum MC placement study by Abts et al. [5] and Xu et al. [8] are no longer valid, when the LLC/directory to MC traffic is considered. Hence, our proposed correction is vital and crucial for emerging NoCs with wireless [12], optical [18] and 3D networks [19].

III. Memory and Cluster Modes in Modern CMPs

A. Memory Modes in Xeon-Phi Architecture

Xeon-Phi architectures have a high-bandwidth MCDRAM memory and a larger low-bandwidth DDR memory [1]. These two memory types can be configured at boot time in different ways, as illustrated in Figure 2.

- **Flat Mode:** In the *flat* mode, both the MCDRAM and DDR memory are mapped in the same system address space. This mode is ideal for applications with data that can be separated into categories of a larger, low-bandwidth region, and a smaller, high-bandwidth region.
- **Cache Mode:** In the *cache* mode, MCDRAM acts as a last level cache which is placed in between the DDR memory and L2 cache. The cache is direct mapped with a cache line size of 64-bytes. All memory requests first go to the MCDRAM for a cache memory lookup, if there is a cache miss, they are sent to the DDR memory.
- **Hybrid Mode:** In the *hybrid* mode, part of MCDRAM (half or quarter) is used in *cache* mode while the rest is used as *flat* mode memory. The DDR memory will be served by the cache portion. This works well for a variety of applications that take advantage of storing frequently accessed data in *flat* memory while also benefiting from regular caching.

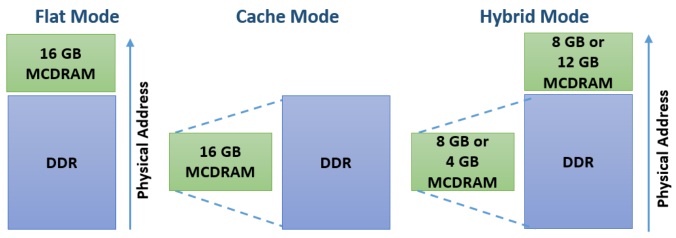


Fig. 2: Three memory modes in Xeon-Phi architectures [1].

B. Cluster Modes in Xeon-Phi Architecture

The mesh interconnect in KNL supports three cluster modes, which have significant impact on the NoC traffic behavior [1]. Similar to memory modes, cluster modes can also be selected from BIOS during boot time.

- **All-to-all mode:** In this mode, there is no affinity between the processing element (PE), MC and directory. That is, a memory request can go from any directory to any MC. As a result, this mode does not exploit locality, unlike the other two cluster modes.
- **Quadrant mode:** In the *quadrant* mode, the chip is divided into four quadrants. There is an affinity between the directories and MC in the same quadrant. However, there is no affinity between the PE and directory, i.e, a processor can send the memory request to any directory, but the directory will always forward that request to an MC on the same quadrant.
- **Sub-NUMA mode:** This mode takes one more step forward by enforcing affinity between all three components - PE, MC and directory. A request from a PE lands on a directory on the same quadrant, and the directory can forward that request to an MC on the same quadrant.

The optimal combination of memory and cluster modes depends on the application characteristics and, largely affects the power and performance statistics.

Figure 3 illustrates the traffic flow of these memory and cluster modes using examples. The *quadrant* and *sub-NUMA* clustering modes improve the locality of memory traffic. For instance, Figure 3c illustrates the *quadrant* mode in KNL [1]. The initial request from a core can go to any directory (1). However, each directory is associated with the MCs within the same quadrant. The memory request marked with (2) can go to integrated MC on the right side or to MCDRAM controllers at the upper right corner. This affinity helps in localizing the directory-memory traffic, which in turn improves memory access latency.

IV. Accurate Modeling of LLC/Directory to Memory Communication

In this section, we first describe how the transactions between core, LLC/directory and memory occur in modern CMPs. Then, we contrast it to the assumption made by gem5 and highlight the consequences. Next, we present an accurate NoC modeling and implementation of cluster and memory modes in gem5. Finally, we demonstrate that our framework is vital to accurately model and explore modern CMPs.

A. Memory Controller Placement in CMPs

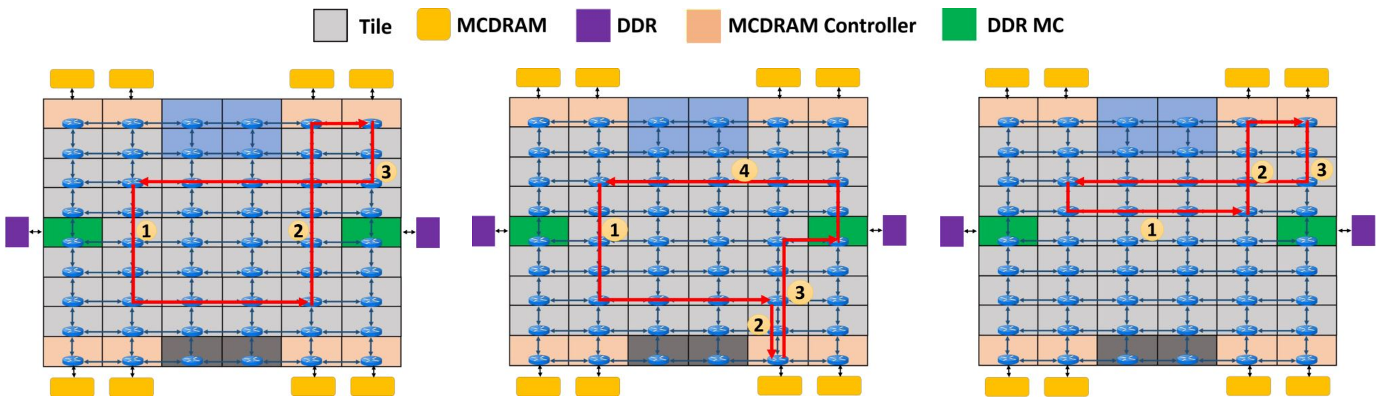
Due to pin limitations and package constraints, it is unrealistic to attach a memory controller to each core in a CMP. For example, Intel Core i7-900 processor has only one MC, and 27.3% of its total pins are dedicated to the MC [20]. Similarly, the Tilera Tile64 processor integrates 64 cores in an 8x8 mesh with four on-chip MCs [3]. This results in a core to MC ratio of 16:1. The total number of cores and memory controllers in several modern CMPs are summarized in Table I.

TABLE I: Comparison of cores and number of MCs in modern many-core CMPs.

Processor	# Cores	# Memory Controllers
Intel Xeon Phi 7210 [21]	64	8 MCDRAM & 2 DDR4
Tilera Tile64 [3]	64	4 DDR2 in 16 ports
Intel Xeon 8160M	24	2 DDR4, 6 channels
AMD Opteron 6386 SE	16	1 DDR3, 4 channels

Several studies have shown that relative placement of cores and MCs plays an important role in network traffic distribution [5], [8]. This impact is more significant in topologies, such as 2D Mesh, which do not have edge symmetry. Thus, it is evident that connecting MCs to every tile gives a highly optimistic estimate of the realistic scenario. Moreover, a large fraction of traffic in a CMP originates not from actual data transfers, but from communication between cores to maintain data coherence [15]. As soon as the directory component comes into play, the traffic distribution is not the same as processor to processor traffic or processor to memory traffic. Therefore, the affinity between the cores, directories and MCs affect the performance of architectures that employ a distributed directory-based cache coherence algorithm. *Consequently, it is crucial to accurately account for the communication flow between the cores, tag directories and memory controllers.*

Arguably, the most widely used architectural simulator - gem5 [6] makes an unrealistic assumption that there is an interface to main memory from every tile of the NoC. This eliminates the exploration of affinity between directory and



(a) Example of L2 miss in *flat* memory mode and *all-to-all* cluster mode: (1) L2 cache miss. Memory request injected on the network to check the tag directory, (2) request forwarded to any memory controller after miss in tag directory, (3) data read from memory and sent to the requester.

(b) Example of L2 and MCDRAM miss in *cache* memory mode and *all-to-all* cluster mode: (1) L2 cache miss. Memory request injected on the network to check the tag directory, (2) request forwarded to MCDRAM which acts as a cache after miss in tag directory, (3) request forwarded to memory after miss in MCDRAM, (4) data read from memory and sent to the requester.

(c) Example of L2 miss in *flat* memory mode and *quadrant* cluster mode: (1) L2 cache miss. Memory request injected on the network to check the tag directory, (2) request forwarded to memory controller on the same quadrant, (3) data read from memory and sent to the requester.

Fig. 3: Traffic models in *flat* and *cache* memory modes and *all-to-all* and *quadrant* cluster modes in KNL architecture [1].

MC. Furthermore, the effects of memory modes cannot be captured in the current gem5 setup.

B. LLC/Directory to Memory Communication in CMPs

A miss in the local cache triggers a sequence of transactions in many-core architectures with distributed directories, as demonstrated Figure 4a. The order of these transactions are as follows:

1) The request is forwarded to the directory controller which contains the memory address information,
2) If data is not available in any of the caches, the request is forwarded to an MC,
3) The data is retrieved from the memory,
4) The MC forwards the data to the requester.

The last two steps are significant, since they introduce many-to-few communication pattern due to the smaller number of MCs, as summarized in Table I. As a result, they not only increase the number of packets in flight, but also lead to hotspots which contribute to increased latency.

C. Unrealistic Assumptions in gem5 on LLC/Directory to Memory Communication

gem5 is one of the most popular many-core architecture simulators [6]. Instead of following these steps given in Section IV-B, it models the memory accesses directly from the directory (home node) itself, as illustrated in Figure 4b. The first step is the same as shown in Figure 4a. That is, the request goes from the core to the tag directory responsible for the corresponding memory address. If there is an LLC miss, the data needs to be fetched from the memory, as expected. However, the memory access is modeled within the home directory (2), without explicitly modeling the traffic from the directory to memory controller. In other words, each "directory controller" implies both a directory (i.e. state) and an MC [6]. The model accounts for the delay to main memory, but it does not have a separate MC node in the NoC. Therefore, the NoC traffic to and from the memory controllers is not modeled. In contrast, the data is forwarded directly from the directory to the requester (3). Comparing the two scenarios, we can see that step 2 in Figure 4a does not exist in the current gem5 model. Moreover, the data (step 3) is sent from the directory, not from the MC as in the realistic model.

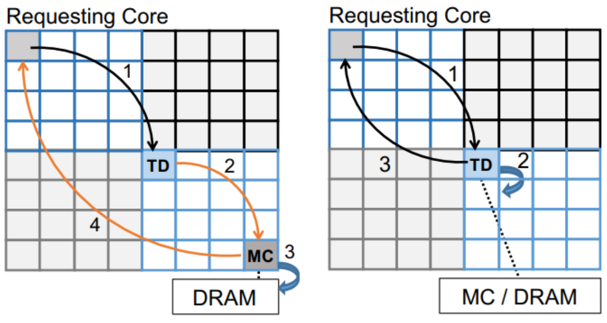


Fig. 4: (a) Life cycle of a memory request and resulting transactions in real distributed directory systems. (b) The same transactions modeled in default gem5. We modified gem5 to have this realistic flow. In the *quadrant* mode, there is an affinity between a tag directory and the MC in that quadrant. Our modification enables to accurately model this architectural feature, while the default model shown in (b) cannot differentiate the affinity.

Impact on NoC Traffic: The modeling choice in Figure 4(b) essentially establishes a virtual link between the tag directory and memory controllers. Therefore, the request and data packets to and from MCs are completely missed in this simulation model. This affects not only the communication latency of a given transaction but also the utilization of the links and routers on the path. Consequently, the latency of *all the NoC traffic* that goes through those routers will be lower in simulation than their actual values. Hence, this will result in optimistic performance estimates.

D. Modeling and Exploration of Intel Xeon-Phi architecture

An accurate NoC simulation model should explicitly capture PE to directory, directory to memory and memory to PE traffic.

Mapping Addresses to Memory Controllers: Since all the cores share the MCs, we need a mechanism to allocate different address ranges to the available MCs. To achieve this, the physical address of a memory location is mapped to an MC according to the function shown in Listing 1. It allocates a certain set of bits from the address to select the MC by defining the range of bits from *small* to *big* and dividing the addresses uniformly among MCs. In this formulation, *addr* is the address to map, *small* is calculated as (*numa_high_bit - num_memories_bits + 1*), and *big* is *numa_high_bit*. Here, *numa_high_bit and num_memories_bits* are calculated depending on the number of MCs. These expressions enable an even distribution of memory addresses among the MCs, which is similar to the decisions in a modern CMP [22].

Listing 1: Address hashing function used to map an address to a memory controller

```
Addr bitSelect(Addr addr, unsigned int small,
    unsigned int big)
{
    assert(big >= small);

    if (big >= ADDRESS_WIDTH - 1) {
        return (addr >> small);
    } else {
        Addr mask = ~((Addr)~0 << (big + 1));
        Addr partial = (addr & mask);
        return (partial >> small);
    }
}
```

Simulation Framework: We employ a cycle-accurate full-system simulator - gem5 [6] and "GARNET2.0" [23] interconnection network model. The default gem5 model is modified to include separate MCs and to model PE to PE, PE to directory, directory to memory as well as memory to PE traffic. The gem5 implementation handles the traffic flow through coherence protocols. In a distributed cache coherence protocol, in case of a cache miss, the request is forwarded to the coherence protocol controller. It makes the necessary state transitions and pushes the message in the appropriate virtual network to the network interface. The network interface then converts the message into network packets and sends them to the network via the connected router. The network then routes the flits to the destination node using X-Y deterministic routing protocol. When the home directory receives the packet, it checks its state machine to see if another cache shares that data. If yes, it forwards the packet to the owner and then to the requestor (PE) and if not, it initiates a memory fetch depending on which memory and cluster modes are being used.

If it is *all-to-all* and *flat* mode, addresses are uniformly distributed across MCDRAM and DDR memory spaces. Which MCDRAM/DDR memory controller to forward to is decided using the function in Listing 1. If it is *quadrant* and *flat* mode, only MCs in that quadrant are considered as candidates for forwarding the memory requests. In *all-to-all* and *cache* mode, MCDRAM space is treated as a last-level cache. Therefore, the request is sent to an MCDRAM controller for a cache lookup. If it is a miss, the memory request is again forwarded to the appropriate MC (selected using Listing 1 without considering MCDRAM controllers), and memory fetch request is placed through there. Once the requested data is fetched from either the DDR or MCDRAM memories, it is forwarded back to the PE after making the necessary coherence transitions.

We explicitly differentiate the behavior of MCDRAM memory in *cache* and *flat* modes. In *cache* mode, MCDRAM cache modules are instantiated and can be accessed only through the designated MCDRAM controller locations. In *flat* mode, this cache module is not used, and an MC similar to the MCs interfacing DDR memory is connected to the designated nodes.

We emphasize that without our modification of the gem5 model, it is not possible to explore the power consumption and performance impact of different cluster and memory modes. The next section highlights two important aspects of our exploration framework. Our proposed NoC model is realistic since the power and performance numbers are comparable with the results from the Xeon-Phi hardware board. Moreover, our framework can be used to accurately model and explore a wide variety of current and future NoC architectures.

V. Experimental Results

A. Experimental Setup

Architecture Model: In our studies, we use the Intel Xeon Phi 7210 platform [21] and model the same on gem5 [6]. It mainly targets high performance computing and other parallel computing segments. The architecture offers high memory bandwidth and massive parallelism options which enables it to run memory and processor intensive workloads with high throughput.

A 64-core CMP is modeled with gem5 using a mesh topology. Each tile is composed of a core that runs at 2 GHz, private L1 cache, tag directory and a router. Each cache is split into data and instruction caches with 16kB capacity each. GARNET2.0 [23], which leverages the routing infrastructure provided by ruby memory system, models a router with a crossbar switch, switch allocation, virtual circuit selection and 4 input buffers giving a 3-cycle pipeline. Each router is connected to four other routers with internal links and to an L1 cache and a directory controller through individual network interfaces via external links. The complete set of simulation parameters are summarized in Table II.

NoC Power Model: Since dynamic power consumption of an NoC is a function of the traffic flow, we need to use an energy model that captures the changes in the traffic flow. We use the model in [24] to estimate the power consumption. According to the energy model, there are two main contributors to NoC power;

1) Number of packets injected into the network - this is directly related to the number of cache misses in L1 and L2 caches, and in *cache* memory mode, misses in MCDRAMs.
2) Average hops traversed by packets - depends on the relative placement of PE, MCs and directories. The affinity between these components which are configured using the cluster modes also contributes to the number of hops.

We feed the output statistics from gem5 to the McPAT power modeling framework [25]. Power consumption of other components - caches, processor, off-chip memory and directories, are estimated using the energy models in McPAT.

TABLE II: System configuration parameters used in our simulations.

Processor Configuration	
Number of cores	64
Core frequency	2 GHz
Instruction set architecture	x86
Memory System Configuration	
L1 cache	private, separate instruction and data cache. Each 16kB in size.
Cache coherence	distributed directory-based protocol
Memory size	4GB DDR
Access latency	300 cycles
Interconnection Network Configuration	
Topology	8x8 Mesh (formed by rings in rows and columns)
Routing scheme	X-Y deterministic
Router	4 port, 4 input buffer router with 3 cycle pipeline delay
Link latency	1 cycle
Parameters that change when implementing KNL	
Number of cores	32 (in 32 tiles each with one core)
Core frequency	1.4 GHz
L1 cache	private, separate instruction and data cache. Each 32kB in size.
MCDRAM	shared, direct mapped cache

Benchmarks: We use benchmarks from SPLASH2 [26] and MiBench [27] benchmark suites to run on gem5.

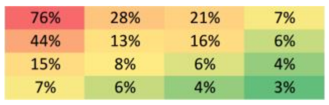
B. *Parameters used to model KNL*

The number of cores in gem5 must be a power of 2. We have 32 tiles with cores similar to KNL. However, each tile contains a single core unlike KNL, since gem5 does not support tiles with two cores. To match the number of cores, we deactivate one core in each tile in our Xeon-Phi platform. We also place the MCs to match the KNL architecture shown in Figure 3. Moreover, we set the core frequency to 1.4 GHz when comparing the simulation results against the hardware measurements to match our Xeon-Phi platform frequency. The parameters used in implementing KNL are summarized in Table II.

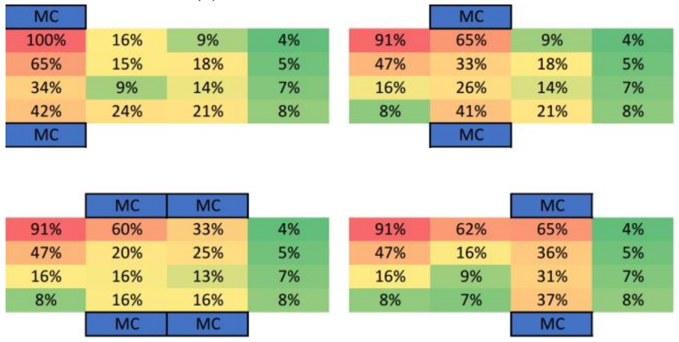
C. *Network traffic analysis of realistic and unrealistic models*

To compare the effects of realistic (proposed approach) and unrealistic (default gem5) models, we observe the buffer utilization at each router as shown in Figure 5. Figure 5a shows a 4x4 mesh where the MCs are connected to each directory which is the default implementation of gem5. Traffic is uniform except for the tile 0 where the PE resides (tile numbers are as shown in Figure 1). Figure 5b shows a realistic scenario where every other parameter is kept the same, but MCs are connected to boundary routers. This does not display the uniform traffic distribution as shown in Figure 5a. Traffic patterns show hotspot columns due to MC placement which increases latency and saturates the throughput. The 4x4 mesh and MC placement configurations used in Figure 5 are for illustration only. Experiments are carried out using the parameters mentioned under section V-A.

As a result of this congestion and more packets being sent through the network, the realistic model shows a 54.9% more network flit latency on average across *FFT, FMM, RADIX and LU* benchmarks compared to the unrealistic model with a similar topology. A comparison of network latency, NoC power usage and execution times with different benchmarks is shown in Figure 6. As stated before, the default gem5 model does not permit cluster mode exploration as MCs are collocated with directories at every tile. Even then, if the default model is used for exploration, the results in Figure 6 show that it gives highly optimistic results for NoC latencies and power.



(a) MCs modeled at each tile.

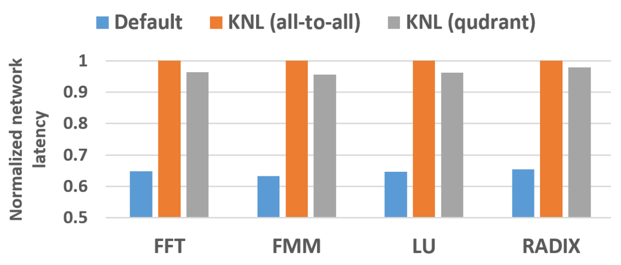


(b) MCs modeled only at places marked in blue.

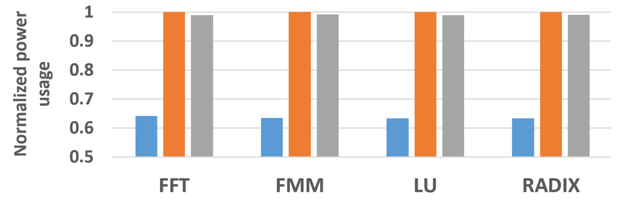
Fig. 5: Buffer utilization in routers when RADIX benchmark running on core 0. Value in each tile is normalized to the highest buffer utilization value in the modified gem5 implementation (Figure 5b). Color coded to show the distribution of utilization across tiles (dark green - lowest and dark red - highest).

D. *Traffic Variation with different cache coherence protocols*

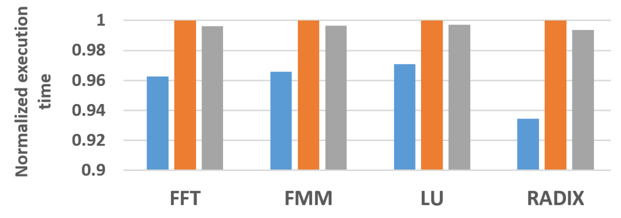
Another factor that effects the NoC traffic behavior is the cache coherence protocol [28]. The default gem5 NoC implementation already captured these variations as it correctly implemented the PE to directory affinity. We explored the effects of different cache coherence protocols - (1) MI, (2) MESI Two-Level, (3) MOESI CMP Directory, (4) MOESI



(a) Normalized network latency.



(b) Normalized NoC power usage.



(c) Normalized execution time.

Fig. 6: Power and performance comparison for different models: a) Default: unrealistic gem5 model which collocates MCs with directories at each tile. b) KNL (*all-to-all*): gem5 KNL model with *all-to-all* cluster mode and *flat* memory mode. c) KNL (*quadrant*): gem5 KNL model with *quadrant* cluster mode and *flat* memory mode.

Hammer [6]. Figure 7 shows traffic variation with different cache coherence protocols when *RADIX* benchmark is running on a 4x4 Mesh NoC. Figure 7a, show buffer utilization at each router when MI cache coherence protocol is used with the default gem5 implementation, which assumes a memory interface at each tile. Similar to the observation of Figure 5a, the traffic shows a uniform gradient across the routers without any congestion. Figures 7b shows the same results with our modified implementation. We observe that the patterns remain consistent across cache coherence protocols. As evident from Figure 7c & 7d, Figure 7e & 7f and Figure 7g & 7h pairs, this observation remains the same across other 3 cache coherence protocols as well. Therefore, the observations made in Section V-C hold irrespective of the cache coherence protocol. This is expected as our modification only affects the affinity between directory and MC. With increased sharing in cache coherence protocols, the traffic in NoC increases. But, the hotspot locations and the traffic distribution remain the same.

E. Network Latency Comparison for different MC placements

Xu et al. explored network traffic behavior with different MC placements (Column 0/7, Column 2/5, Diamond, Slash and Optimal) and concluded that the "Optimal" was best for similar benchmarks [8]. Figure 8 shows network flit latency

83%	38%	25%	13%
40%	15%	13%	10%
26%	13%	10%	8%
13%	10%	8%	5%

(a) MI

	MC	MC	
100%	81%	41%	7%
44%	23%	20%	9%
27%	26%	21%	11%
13%	28%	28%	14%
	MC	MC	

(b) MI

80%	37%	25%	13%
39%	16%	13%	11%
26%	13%	11%	8%
14%	11%	9%	6%

(c) MESI Two Level

	MC	MC	
100%	81%	42%	8%
44%	25%	21%	10%
29%	28%	23%	13%
15%	31%	33%	16%
	MC	MC	

(d) MESI Two Level

83%	36%	25%	13%
43%	16%	13%	11%
29%	14%	11%	8%
15%	12%	9%	6%

(e) MOESI CMP Dir.

	MC	MC	
100%	77%	39%	7%
46%	25%	20%	10%
30%	29%	23%	13%
16%	33%	34%	17%
	MC	MC	

(f) MOESI CMP Dir.

66%	46%	33%	18%
29%	22%	21%	17%
22%	19%	18%	15%
12%	14%	13%	9%

(g) MOESI Hammer

	MC	MC	
83%	100%	55%	11%
33%	34%	32%	17%
24%	40%	37%	23%
14%	38%	49%	26%
	MC	MC	

(h) MOESI Hammer

Fig. 7: Buffer utilization in routers when RADIX benchmark is running on a 4x4 Mesh with different cache coherence protocols. The color code is similar to what is used in Figure 5. (a), (c), (e), (g) show results with default gem5 implementation and (b), (d), (f), (h) show the same after our modification. The buffer utilization in each tile is normalized to the highest value in the modified gem5 implementation for a given cache coherence protocol.

when realistic MC placement configurations in [8] as well as gem5 default (unrealistic) model are tested across different benchmarks. As further evidence for the highly optimistic nature of the default gem5 model, we can see that the latency is significantly less when compared to realistic MC placement models.

In contrast to the conclusion in [8], "Optimal" is no longer the best placement, *when the directory-based coherence is introduced*. The results not only depend on MC placement, but also on PE placement and coherence protocol. Considering only the realistic placements described in [8], Column 2/5 configuration turns out to be the best by 9.0% compared to the worst configuration (Slash) when running *BASICMATH*. Column 2/5 also beats "Optimal" by 5.3% on average across all benchmarks. The traffic congestion caused by adjacent MCs in Column 2/5 configuration is compensated by the reduced hop counts, since it gives smallest average hop count.

F. Exploration of memory and cluster modes and validation with results from the KNL hardware platform

As seen from results in Figure 8, the affinity between the PE, MC and directory plays a major role in network traffic behavior. To explore this further, we experimented with

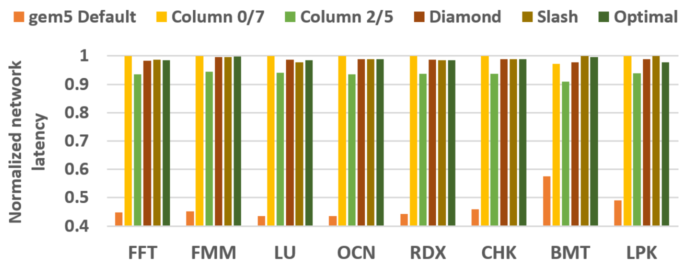


Fig. 8: Normalized network latency with different MC placements. FFT, FMM, LU, OCEAN (OCN), RADIX (RDX), CHOLESKY (CHK) from SPLASH2, and BASICMATH (BMT), LINPACK (LPK) from MiBench are used as benchmarks.

different cluster and memory modes available in the KNL architecture and validated the simulation results with Intel Xeon Phi 7210 platform. The results for both *all-to-all* and *quadrant* cluster modes as well as *flat* and *cache* memory modes are shown in Figure 9. Compared to *all-to-all flat* mode, *all-to-all cache* mode gives the highest benefit with 18.62% less execution time on average across all benchmarks. That is an average speedup of 1.23. The average speedup of *quadrant flat* mode over *all-to-all flat* is small (1.013). This is mainly because the benchmarks do not stress the platform too much and memory access latency hinders the savings of network flit latency. These observations are in agreement with the Intel Xeon Phi results [1], which further justifies the accuracy of our approach. Clearly, it is not possible to perform these memory and cluster mode explorations without the proposed NoC modeling framework.

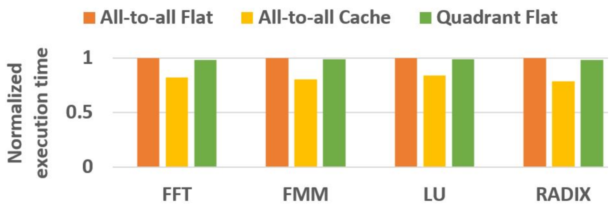


Fig. 9: Normalized execution times with KNL architecture modeled in gem5. Baseline execution times are as follows: FFT (2.99 seconds), FMM (0.05 seconds), LU (0.02 seconds) and RADIX (0.18 seconds).

VI. Conclusion

In this paper, we explored how the network traffic behaves when directory-based cache coherence is introduced. The change in traffic behavior is not captured in the widely used gem5 simulator and thus it can lead to unrealistic conclusions. This paper made three important contributions. First, we observed that the gem5 model is faulty as it models an MC at every tile, thus eliminating coherence traffic and also not practical due to pin limitations. Next, we implemented an accurate and realistic model to explore MC placement in an 8x8 mesh with 16 MCs. Our results show that the previous conclusions made without considering coherence traffic are no longer valid. Our studies show that optimization methods should not only consider MC placements, but also PE placement and coherence protocol to come to realistic conclusions. Finally, our experimental results demonstrated that the affinity between MC and directory controller can be manipulated with different cluster and memory modes such as *quadrant*, *all-to-all*, *flat* and *cache* modes introduced in Intel's KNL architecture to achieve better performance and power results. Our proposed exploration framework is vital for emerging NoCs with wireless, optical and 3D networks.

References

[1] A. Sodani *et al.*, "Knights landing: Second-generation Intel Xeon Phi product," *IEEE Micro*, vol. 36, no. 2, pp. 34–46, 2016.
[2] Y. Hoskote *et al.*, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.
[3] D. Wentzlaff *et al.*, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, 2007.
[4] U. Y. Ogras and R. Marculescu, *Modeling, analysis and optimization of network-on-chip communication architectures*. Springer, 2013.
[5] D. Abts *et al.*, "Achieving predictable performance through better memory controller placement in many-core cmps," *CA News*, vol. 37, no. 3, pp. 451–461, 2009.
[6] N. Binkert *et al.*, "The gem5 simulator," *CA News*, vol. 39, no. 2, pp. 1–7, 2011.
[7] J. W. Lee *et al.*, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *CA News*, vol. 36, no. 3, 2008, pp. 89–100.
[8] T. C. Xu *et al.*, "Optimal memory controller placement for chip multiprocessor," in *CODES+ISSS*, 2011, pp. 217–226.
[9] J. Howard *et al.*, "A 48-core ia-32 message-passing processor with dvfs in 45nm cmos," in *ISSCC*. IEEE, 2010, pp. 108–109.
[10] P. Eitschberger *et al.*, "Exploring the placement of memory controllers on manycore processors: A case study for Intel SCC," in *MCC*, 2013.
[11] M. Awasthi *et al.*, "Handling the problems and opportunities posed by multiple on-chip memory controllers," in *PACT*, 2010, pp. 319–330.
[12] K. Duraisamy *et al.*, "Multicast-aware high-performance wireless network-on-chip architectures," *IEEE TVLSI*, vol. 25, no. 3, pp. 1126–1139, 2017.
[13] P. Conway and B. Hughes, "The AMD Opteron northbridge architecture," *IEEE Micro*, vol. 27, no. 2, 2007.
[14] A. Ros *et al.*, "Dealing with traffic-area trade-off in direct coherence protocols for many-core CMPs." in *APPT*. Springer, 2009, pp. 11–27.
[15] M. Schuchhardt *et al.*, "The impact of dynamic directories on multicore interconnects," *Computer*, vol. 46, no. 10, pp. 32–39, 2013.
[16] B. A. Cuesta *et al.*, "Increasing the effectiveness of directory caches by deactivating coherence for private memory blocks," in *CA News*, vol. 39, no. 3, 2011, pp. 93–104.
[17] J. Zebchuk *et al.*, "A tagless coherence directory," in *MICRO*. ACM, 2009, pp. 423–434.
[18] S. V. R. Chittamuru *et al.*, "Swiftnoc: a reconfigurable silicon-photonic network with multicast-enabled channel sharing for multicore architectures," *JETC*, vol. 13, no. 4, p. 58, 2017.
[19] R. Morris *et al.*, "3D-NoC: Reconfigurable 3d photonic on-chip interconnect for multicores," in *ICCD*, 2012, pp. 413–418.
[20] "http://download.intel.com/design/processor/datashts/320834.pdf," Intel Core i7-900 Processor.
[21] "http://ark.intel.com/products/94033/Intel-Xeon-Phi-Processor-7210-16GB-1_30-GHz-64-core," Intel Xeon Phi Processor 7210.
[22] Y. Kim *et al.*, "Atlas: A scalable and high-performance scheduling algorithm for multiple memory controllers," in *HPCA*, 2010, pp. 1–12.
[23] N. Agarwal *et al.*, "Garnet: A detailed on-chip network model inside a full-system simulator," in *ISPASS*. IEEE, 2009, pp. 33–42.
[24] U. Y. Ogras *et al.*, "Design and management of voltage-frequency island partitioned networks-on-chip," *IEEE TVLSI*, vol. 17, no. 3, 2009.
[25] S. Li *et al.*, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.
[26] S. C. Woo *et al.*, "The splash-2 programs: Characterization and methodological considerations," in *Computer Architecture*, 1995.
[27] M. R. Guthaus *et al.*, "Mibench: A free, commercially representative embedded benchmark suite," in *WWC*, 2001.
[28] D. Molka *et al.*, "Cache coherence protocol and memory performance of the intel haswell-ep architecture," in *ICPP*. IEEE, 2015, pp. 739–748.