Memory Network: Enabling Technology for Scalable Near-Data Computing

Gwangsun Kim, John Kim
KAIST
{gskim, jjk12}@kaist.ac.kr

Jung Ho Ahn
Seoul National University
gajh@snu.ac.kr

Yongkee Kwon
SK Hynix
yongkee.kwon@sk.com

1. INTRODUCTION

Processing-in-memory (PIM) or near-data processing (NDP) has been researched during the late 90s but with the advances in technology, including 3D memory stacking, and new emerging workloads, similar concepts have been recently revisited. There are various challenges for near-data processing to become widely used but one of the basic assumption is the emphasis on *near*-data processing – i.e., the data is "near" the computation. For a single-node PIM or NDP, this is not an issue but if there are more than one nodes in the system, a challenge is that the data might not be "near" any longer, especially if the data is located on a different node. As a result, to enable scalable near-data processing, one of the challenges includes how to efficiently move data between the various nodes or minimize the cost of data movement.

In this paper, we describe the issues in designing a cost-efficient *memory network* that can enable scalable near-data processing. In particular, we focus on multi-module PIM or NDP system – i.e., a system that has more than one PIM module. With such systems, we describe the *memory network* or the interconnect that connects the memory modules and the core modules together. While there are high-level, software challenges including data mapping, and memory allocation in NDP, the underlying communication substrate (or the memory network) is a critical component in enabling efficient communication and scalable near-data computing. We describe some of the recent work in memory network and then, summarize different directions for research in memory network for scalable near-data computing.

In this work, we assume a hybrid memory cube (HMC)-based NDP as shown in Fig. 1, but this work is not necessarily limited to the HMC technology and is applicable to other memory technologies. Processors and the HMCs communicate with packetized abstract messages through high-speed serial interface instead of communicating low-level commands (e.g., DDR memory commands). The memory is 3D stacked on top of a logic layer, connected with through-silicon-vias. The logic layer consists of the vault memory controllers and the I/O ports. In addition, an *intra*-HMC network connects the I/O ports and the memory controllers together.

Prior work [5] has classified the different types of interconnects with multiple memory and processing modules. Conventional multi-socket systems can be classified as *processor*-centric network (PCN) as the processor-interconnect is used to connect the different processors

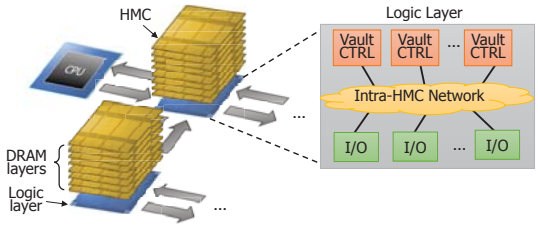
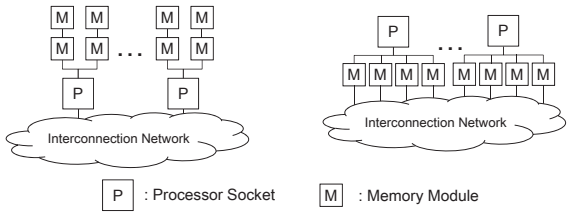


Figure 1: The Hybrid Memory Cube (HMC) architecture.



(a) Processor-centric network (b) Memory-centric network

Figure 2: Alternative system interconnect designs.

together, with each processor having its own dedicated memory, as shown in Fig. 2(a). In comparison, if the memory modules or a pool of memory modules are interconnected with a memory network, a *memory*-centric network (MCN) can be created. In an MCN, processors are only connected to their local HMCs, and all communications between processors are routed through the memory network as shown in Fig. 2(b) [5]. While MCN enables flexibility and facilitates remote memory access, it also creates challenges since locality can be lost if accessing data from remote memory modules. In this work, we focus on the memory-centric network organization or simply, memory network [10] and the challenges/issues.

1.1 Related Work

When PIM was researched in the late 90s to early 2000s, PIM interconnect or memory interconnect was also discussed. For example, in DIVA [3], the interconnect design to connect multiple PIM modules was presented. In particular, the authors designed a simple, fast router organization but assumed a simple low-dimensional network (e.g., ring) was sufficient. However, with the advances in technology and greater importance on energy-efficiency, such an approach is not necessarily most efficient as higher hop count results in inefficiency – both in terms of performance and energy.

With the recent (re-)emergence of PIM or near-data processing, the interconnect aspect of PIM has been revisited as well. Sudan [13] and Rosenfeld [12] have eval-

uated alternative memory network topologies including daisy chain organization [12] and different tree organizations [13]. However, their work was focused on a single-node or a single-socket system with multiple memory modules. With only a single-node system, a "halo"-like organization can be most efficient to minimize distance between the node and the memory modules.

Different topologies for multi-CPU systems [5] and multi-GPU systems [6] have been recently proposed. Their topologies are variations of the recently proposed high-radix topologies for large-scale networks, including the flattened butterfly topology [7] and the dragonfly topology [8]. These topologies differ from other work as multiple processing nodes are assumed in the memory network. To exploit the constraints of a memory network, these topologies exploit *distributor*-based topology to efficiently utilize the channel bandwidth from the processing nodes. In the following section, we summarize and highlight some of the issues in designing such memory networks and the different research directions.

2. MEMORY NETWORK

2.1 Common Interface

One of the challenges (as well as one of the significant benefits) of the memory network comes from a common interface across all of the modules. Current HMC is based on an abstract interface between the processing nodes and the 3D stacked memory where packets are used for communication – a very different interface compared with the conventional DRAM interface. There is a non-technical challenge associated with this issue since multiple vendors need to agree on a common interface but a recent whitepaper argues that "computer industry has historically made major advances when industry players have been able to add innovation behind a standard interface" [11].

By having a common interface, it not only results in greater flexibility in system configuration but it can also potentially reduce overall cost of the system. For example, in a heterogeneous system such as a multi-GPU system today, various interconnects are necessary – including the CPU memory bus, the GPU memory bus, and the PCIe interconnect for communication between the CPU and the GPUs. A memory-network for such a heterogeneous system can be built with just a single type of interconnect and reduce the system cost.

With a packetized approach in the interface and creating an interconnection network, the significant amount of research done in large-scale networks (and on-chip networks) can be leveraged. However, some of the problems unique to the memory network need to be addressed. For example, with multiple packets in the network, there is no guarantee of ordering, if that is required, in the system leveraging memory network. In addition, there is no bound (or guarantee) on the latency of the packets injected into the network (compared with a request sent through a conventional DRAM bus interface). These challenges must be addressed to enable a memory network using a common, packetized interface.

2.2 Topology

While a daisy-chain approach is a simple topology, it is not necessarily an efficient memory network topology since each additional hop increases network latency and power. Designing a high-radix topology that reduces the network diameter and thus, latency and power, is important; however, creating a high-radix topology in a memory network is a challenge since the number of router ports in an HMC is relatively small (e.g., current HMC generation has 8 ports). The Dragonfly [8] is a topology that creates *virtual* high-radix router by using a collection of routers. This can help reduce diameter by virtually increasing the radix but it comes at the cost of increasing local hop count. While previously proposed high-radix topologies are an *indirect* network, the memory network topology needs to be a *direct* network since the routers are directly connected to the endpoint nodes (e.g., memory modules). As a result, in addition to high-radix topologies, a topology exploration is necessary to determine the most efficient topology for scalable memory network topology.

In addition to the topology itself, the routing also has significant impact on overall performance. One of the advantages of a high-radix topology is the path diversity it provides and the routing algorithm needs to fully exploit it. However, providing more path diversity than necessary leads to increased cost — for example, if well load-balanced memory mapping is used (e.g., fine-grain cache line interleaving), it can lead to random traffic and remove the need for any path diversity or adaptive routing. However, this results in loss of locality as the data is not necessarily nearby. Thus, a balance between locality and load-balancing needs to be achieved, not only through the topology/routing but also based on the memory access pattern and memory mapping.

2.3 Hierarchical Network

While there is the *inter*-HMC network, there is another intra-HMC network for a network within the logic layer, as shown earlier in Fig. 1. In the design of the intra-HMC network, the inter-HMC network needs to be considered to be able to fully optimize the network design and overall efficiency. Similar approaches have been proposed for large-scale network router design [1] and the Anton2 network [14], referred to as a *unified* network, combines the design of the off-chip network router and the on-chip network. Similarly, the memory network is a hierarchical network and the impact of the local (or the intra-HMC) network and the global (or the inter-HMC) network, and their interaction need to be carefully understood. One example of this is the pass-through architecture [5] where the placement of the I/O ports was based on the overall inter-HMC network topology. This prior work assumed a simple concentrated mesh topology [2] for the intra-HMC network but it remains to be seen what the most efficient intra-HMC network organization is. In addition, although the network size of the intra-HMC network can be relatively small (e.g., 8-16 components), the components are distributed across the entire logic layer chip and thus,

even if a simple crossbar is used, routing the wires to/from the different components and providing global arbitration across the different distributed components is non-trivial.

2.4 Power Efficiency

Power (or energy) efficiency is an important factor in overall system design. For example, one of the limitations of previously proposed Fully Buffered DIMM technology was the high energy cost of routing through intermediate buffers. Thus, if the energy cost of HMC or other similar technology with near-data computing is not properly managed, it will not lead to a scalable solution. One of the challenges with high-speed signaling interface is the power consumed in the channels (and the I/Os). Since the high-speed Ser/Des (serializer/deserializer) consumes significant amount of static power, different power saving techniques are necessary to improve power-efficiency. There are the traditional trade-off issues in power-gating – when to power-gate? which channels to turn off? power-gate latency vs. overhead trade-off, etc. For memory-latency sensitive workloads that do not have high memory throughput requirements, the power-gating strategies will likely be different from high-throughput workloads that require high bandwidth. In addition, the power-gating should not be considered separately but holistically with the topology and the routing. For example, if a channel consumes significant amount of static power, does it make sense to leverage non-minimal routing and keep some of the minimal channel power-gated to improve overall system energy efficiency? The power challenges creates an opportunity for *dynamic* topologies where the topology connectivity changes based on the traffic load.

2.5 Memory Management

NUMA systems often have local memory for a given node (i.e., memory directly connected to the current node) and remote memory. For these systems, there is added cost in accessing remote memory and there has been significant research done on memory management in NUMA systems, including memory data allocation and migration. The same issues are applicable to a memory network but the constraints are different – instead of a physical distinction between local and remote memory, the memory network creates a memory pool [9] that is shared by the computing nodes. As a result, the distinction between local and remote memory is no longer necessarily required; however, optimizing data allocation can have a significant impact on overall memory network traffic and overall performance. The memory network also presents an opportunity for new optimization on page migration.

The ability to share memory through the memory network can be more significant for heterogeneous systems with different components, each with its own local memory. For example, in current multi-GPU systems, the CPU and the GPU have their own memory and communication is necessary to transfer data between the two memory – data initialized on the CPU memory are often copied over to the GPU memory before the GPU begins its computation. This communication can also be a performance bottleneck. By having a common memory network shared between the CPU and the GPUs, such explicit memory copy is no longer necessary, which provides improvement in performance and cost. In addition, accessing remote GPUs data is simplified since only the memory network needs to be accessed and the requests do not need to be routed through remote GPUs. However, memory network between the heterogeneous components also creates other design issues that need to be addressed – for example, how to partition the total physical memory across the different components, as well as how to allocate memory across the different components to exploit locality.

3. SUMMARY

In this paper, we describe the issues in designing a memory network for scalable near-data computing. Although we focused on the HMC technology, there are some limitations currently. For example, some of the basic features required to design an interconnection network (e.g., VCs) are not supported in the current HMC specification [4] and the number of HMCs that can be chained together is also limited (8 HMCs in the current spec). However, there is no fundamental reason why VCs cannot be supported; nor with an efficient, scalable memory network, the number of components does not need to be limited by introducing additional bits in the header to address the different nodes. Research into memory network needs to overlook some of the limitations that might exist in current technologies and explore cost-efficient memory network to enable scalable near-data processing in the future.

4. REFERENCES

[1] J. Ahn *et al.*, "Network within a network approach to create a scalable high-radix router microarchitecture," in *HPCA'12*.

[2] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *ICS'06*.

[3] J. Draper *et al.*, "The architecture of the diva processing-in-memory chip," in *ICS'02*.

[4] "Hybrid Memory Cube Specification 1.0," Hybrid Memory Cube Consortium, 2013.

[5] G. Kim *et al.*, "Memory-centric system interconnect design with hybrid memory cubes," in *PACT'13*.

[6] G. Kim *et al.*, "Multi-GPU system design with memory networks," in *MICRO'14*.

[7] J. Kim *et al.*, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *ISCA'07*.

[8] J. Kim *et al.*, "Cost-efficient dragonfly topology for large-scale systems," *IEEE Micro*, 2009.

[9] K. Lim *et al.*, "Disaggregated memory for expansion and sharing in blade servers," in *ISCA'09*.

[10] D. R. Resnick, "Memory Network Methods, Apparatus, and Systems," US Patent US20100211721 A1, 2010.

[11] D. R. Resnick and M. Ignatowski, "Proposing an abstracted interface and protocol for computer systems," White Paper, Sandia National Laboratories, 2014.

[12] P. Rosenfeld, "Performance exploration of the hybrid memory cube," Ph.D. dissertation, University of Maryland, 2014.

[13] K. Sudan, "Data Placement for Efficient Main Memory Access," Ph.D. dissertation, University of Utah, 2013.

[14] B. Towles *et al.*, "Unifying on-chip and inter-node switching within the anton 2 network," in *ISCA'14*.