Achieving Memory Access Equalization via Round-trip Routing Latency Prediction in 3D Many-core NoCs

Xiaowen Chen[†,‡], Zhonghai Lu[‡], Yang Li[¶], Axel Jantsch[§], Xueqian Zhao[‡], Shuming Chen[†], Yang Guo[†], Zonglin Liu[†], Jianzhuang Lu[†], Jianghua Wan[†], Shuwei Sun[†], Shenggang Chen[†], Hu Chen[†]
[†]College of Computer, National University of Defense Technology, 410073, Changsha, China
[‡]Department of Electronic Systems, KTH Royal Institute of Technology, 16440 Kista, Stockholm, Sweden
[§]Institute of Computer Technology, Vienna University of Technology, 1040 Vienna, Austria
[¶]College of Computer, National University of Defense Technology, 410073, Jilin, China

***Abstract*—3D many-core NoCs are emerging architectures for future high-performance single chips due to its integration of many processor cores and memories by stacking multiple layers. In such architecture, because processor cores and memories reside in different locations (center, corner, edge, etc.), memory accesses behave differently due to their different communication distances, and the performance (latency) gap of different memory accesses becomes larger as the network size is scaled up. This phenomenon may lead to very high latencies suffered from by some memory accesses, thus degrading the system performance. To achieve high performance, it is crucial to reduce the number of memory accesses with very high latencies. However, this should be done with care since shortening the latency of one memory access can worsen the latency of another as a result of shared network resources. Therefore, the goal should focus on narrowing the latency difference of memory accesses. In the paper, we address the goal by proposing to prioritize the memory access packets based on predicting the round-trip routing latencies of memory accesses. The communication distance and the number of the occupied items in the buffers in the remaining routing path are used to predict the round-trip latency of a memory access. The predicted round-trip routing latency is used as the base to arbitrate the memory access packets so that the memory access with potential high latency can be transferred as early and fast as possible, thus equalizing the memory access latencies as much as possible. Experiments with varied network sizes and packet injection rates prove that our approach can achieve the goal of memory access equalization and outperforms the classic round-robin arbitration in terms of maximum latency, average latency, and LSD[1]. In the experiments, the maximum improvement of the maximum latency, the average latency and the LSD are 80%, 14%, and 45% respectively.**

I. Introduction

3D die stacking technology is an emerging solution for future high-performance single chip design, because it continuously increases the number of processor cores and memories by stacking multiple processor layers and memory layers, and reduces the long wire latency and improves the memory bandwidth by using massive TSVs (Through Silicon Vias). Such many processor cores and memories are connected by pipelined communication networks (called Network-on-Chip (NoC)[1]) rather than buses, so hundreds or thousands of communications can go on concurrently at any time. Such architecture is referred to as 3D many-core NoC, which has attracted great attentions. For instance, Kim, et. al. proposed a 3D multi-core system with one 64-core layer and one 256KB SRAM layer using Tezzaron TSV/bonding technology[2]. Fick, et. al. designed a low-power 64-core system, named Centip3De, which has two stacked dies with a layer of 64 ARM Cortex-M3 cores and a cache layer[3]. Furthermore, they extended Centip3De to be a 7-layer 3D many-core system including 2 processor layers with 128 cores in total, 2 Cache/SRAM layers, and 3 DRAM layers[4]. Wordeman, et. al. proposed a prototype of a 3D system with a memory layer and a processor-like logic layer[5]. Yuang, et. al. proposed a 3D mesh NoC, which tightly mixes memories and processor cores so as to improve the memory access performance[6]. All of these researches use NoC as the communication infrastructure and TSV to connect multiple layers.

In 3D many-core NoCs, since processor cores and memories reside in different locations (center, corner, edge, etc.), the memories are asymmetric so as to be a kind of NUMA (Non Uniform Memory Access)[7][8] architecture and memory accesses behave differently due to their different communication distances. As the network size is scaled up and the number of processor cores and memories increases, the communication distance difference of memory accesses becomes larger and a large increased number of memory accesses worsen the network contention and congestion, thus the performance (latency) gap of different memory accesses becomes larger. This phenomenon may lead to very high latencies suffered from by some memory accesses that are the bottleneck of the system and even extremely degrade the system performance. To achieve high performance in emerging 3D many-core NoCs, it is crucial to reduce the number of memory accesses with very high latencies. However, this should be done with care as shortening one memory access's latency can worsen the latency of another because the network resources such as router buffers and communication links are shared. Therefore, the goal should focus on balancing the latencies of memory accesses (i.e. narrowing the latency difference of memory accesses) as well as ensuring a low average latency value, which is referred to as the "memory access equalization" problem in 3D many-core NoCs.

In the paper, we address the goal by proposing to prioritize the memory access packets on the basis of predicting the round-trip routing latencies of memory accesses in the network. A round-trip routing latency of a memory access is its elapsed time in the network, which is equal to the time of the memory request (read or write) routing in the outward trip from the source to the destination plus the time of the corresponding memory response (read data or write acknowledgement) routing back to the source from the destination in the return trip. The basic idea of our approach is that the memory access with longer round-trip routing latency should go first when it competes with other memory accesses. The round-trip routing latency includes the elapsed routing time and the future routing time, which is unknown when a memory access is traversing in the network and hence requires prediction. We use the communication distance and the number of the occupied items in the buffers in the remaining routing path to predict the

[1]LSD: Latency Standard Deviation measures the amount of variation or dispersion from the average latency. A low standard deviation indicates that the latencies tend to be very close to the mean. Therefore, LSD is suitable to evaluate the memory access equalization.

future routing time of a memory access on-the-fly. Experiments demonstrate that our approach can achieve the goal of memory access equalization and has performance improvement in terms of maximum latency, average latency, and LSD in comparison to the classic round-robin arbitration.

The rest of the paper is organized as follows. Section II discusses the related work. Section III details our approach containing the target architecture, motivation, and router design supporting the round-trip routing latency prediction of memory accesses. Section IV reports the experimental results and the performance analysis. Finally, we conclude in Section V.

II. Related Work

For memory access equalization, prior work first studied it in the context of computer systems[9], then some literatures turned their eyes to the chip and paid attentions to studying the equalization of memory accesses to the off-chip SDRAM, because off-chip SDRAM has high capacity and serves a large number of memory accesses so that it is a hotspot/high congestion region with possible heavy contention and the latencies of memory accesses may be so different. Usually, the equalization of memory accesses to the off-chip SDRAM is studied in two aspects: balancing the memory access performance in memory interface to the off-chip SDRAM[10][11][12] or on-chip routers[13]. For the first aspect, In [10], targeting Chip Multiprocessors, Mutlu, et al. presented a fair memory access scheduling mechanism in SDRAM interface, which balances the ratio of memory access latency between shared and alone cases in memory access latency among cores. According to NoC-based multicores, some literatures utilized the on-chip network information to perform fair scheduling of the memory requests in SDRAM interface so as to achieve memory access equalization. For instance, Both [11] and [12] proposed to prioritize requests in SDRAM memory interfaces according to the congestion information such that requests from less-congested regions prioritize over the other requests. The definition and the concrete implementation of their congestion information are different. For the second aspect, memory requests are considered to be scheduled during their transmission in on-chip networks in order to avoid large congestion near the SDRAM interface so that the memory access latency would be more balanced [13]. As the memory size in a single chip increases, on-chip memory access performance gradually attracts researchers. In [14], Pimpalkhute et. al. proposed a holistic solution for intelligently scheduling network packets (on-chip cache requests) and memory packets (off-chip requests) to optimize overall system performance. They balance the latency performance between the on-chip requests and off-chip requests. Different from them, we focus on balancing the latencies amongst the on-chip memory accesses. In [15], Sharifi et. al. addressed balancing latencies of memory accesses issued by an application in an execution phase. They prioritized memory response messages such that, in a given period of time, messages of an application that experience higher latencies than the average message latency of that application are expedited. However, they divided a whole memory access into two parts: outward trip for memory request (read or write) and return trip for memory response (read data or write acknowledgement) and treated them separately. Their scheduling scheme only prioritizes the memory response messages and balances the return trip latencies. Different from them, our approach considers the two parts of an memory access as a whole and the round-trip routing latency is used to prioritize memory accesses so as to achieve the goal of memory access equalization, since considering the entire round trip of a memory access is more reasonable. Besides, to the best of our knowledge, there is little work studying the memory access equalization in the context of 3D many-core NoCs.

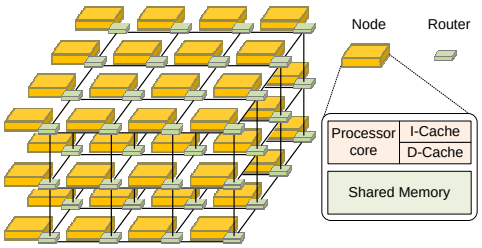

Fig. 1. A homogenous 4×4×3 3D many-core NoC

III. Our Approach

A. Target Architecture: 3D many-core NoCs

Fig. 1 shows an example of our target architecture: homogenous 3D many-core NoCs. The example system is composed of 4×4×3 nodes interconnected via a packet-switched 3D mesh network. Each node is connected to a router. Routers are interconnected with bidirectional links. In homogenous 3D many-core NoCs, all nodes are identical and so are the routers. A node contains a processor core and a shared memory that is visible to all nodes. Memories are distributed and shared so that the centralized memory organization and hence the hotspot area are avoided.

B. Motivation

In such architecture shown in Fig. 1, a large number of memory accesses are generated. Memory accesses behave differently due to their different communication distances. For instance, the up left corner node in the top layer accesses the shared memory in its neighboring node in 2 hops for a round-trip memory read, but it takes 16 hops over the network if it reads a data in the shared memory of the bottom right corner node in the bottom layer. The latency difference results in unequal memory access and some memory accesses with very high latencies, thus negatively affecting the system performance. The impact becomes worse when the network size is scaled up, because the latency gap of different memory accesses becomes bigger. Because memory accesses traverse and contend in the network, improving one memory access's latency can worsen the latency of another. Therefore, we are motivated to focus on memory access equalization in 3D many-core NoCs through balancing the latencies of memory accesses (i.e. narrowing the latency difference of memory accesses) while ensuring a low average latency value.

An entire memory access is a round-trip one containing two parts: memory request (read or write) in outward trip and memory response (read data or write acknowledgement) in return trip, so the performance of a memory access includes that of the two parts. We envision that, to achieve the goal of memory access equalization, it is better to consider the total routing latency of the round-trip of a memory access as the base to prioritize the memory accesses when they contend in the routers (i.e. the memory access with longer round-trip routing latency gains the link successfully and go first), because the round-trip routing latency represents the full performance of a memory access and the two parts should not be considered separately.

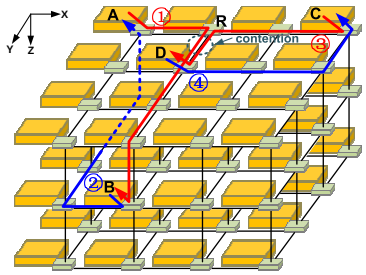

Fig. 2. A short motivation example: two memory accesses traverse and contend.

Let us take Fig. 2 as an example. Fig. 2 shows a 4×4×3 3D mesh NoC, which is packet-switched, performs deterministic DOR[2] X-Y-Z routing and takes one cycle for one hop. In the figure, Node A starts a memory access to the shared memory in Node B, which contains two parts: the memory request from Node A to Node B (the red line ①) and the memory response from Node B to Node A (the blue line②). Meanwhile, a memory access is started from Node C to the shared memory in Node D, which contains two parts: the memory request from Node D to Node C (the red line ③) and the memory response from Node C to Node D (the blue line ④). Assume that memory request ① and ③ contend at router R for the downstream link and memory request ③ occupies the link successfully, therefore memory request ① has to wait one cycle in router R due to its failure. Finally, memory access (A-B) takes 6 cycles[3] for its outward trip and 5 cycles for its return trip, thus 11 cycles in total, while memory access (C-D) takes 6 cycles for its round trip. The average latency of the two memory accesses is 8.5 cycles, and both the two memory accesses deviate from the average latency by 2.5 cycles. Actually, when at router R, memory access (A-B) has taken 1 cycles and memory access (C-D) has used 2 cycles. Although the elapsed time of memory access (A-B) is less than that of memory access (C-D), the former's total time is bigger than the latter's. if the total time (round-trip routing latency) is used as the base for arbitration, memory request ① obtains the link of router R successfully. Therefore, memory access (A-B) takes 10 cycles, while memory access (C-D) takes 7 cycles with 1-cycle waiting time at router R. Although the average latency is still 8.5 cycles, the deviation of both memory accesses from the average latency becomes smaller to be 1.5 cycle. Thus, the difference of the two memory accesses is narrowed and memory access equalization is achieved. Fig. 2 is an intrinsic case to describing our motivation and idea. As we know, when a memory access is being routed in the network, the past routing time is known but the future routing time is unknown, so the round-trip latency (equaling the past routing time plus the future routing time) is needed to be predicted. The next subsection describes our router design supporting round-trip routing latency prediction of memory accesses in detail.

C. Router Design Supporting Memory Access Equalization

The communication infrastructure in our target architecture is a packet-switched 3D mesh network with deterministic DOR X-Y-Z routing, thus preventing cyclic dependencies and avoiding network deadlock. Memory access equalization is supported in routers during the transmission of memory access packets in the network.

(I) Router Microarchitecture

[2]Dimension-Ordered Routing

[3]For calculation simplicity, only the time elapsed in the network rather than in the node is considered in the example.

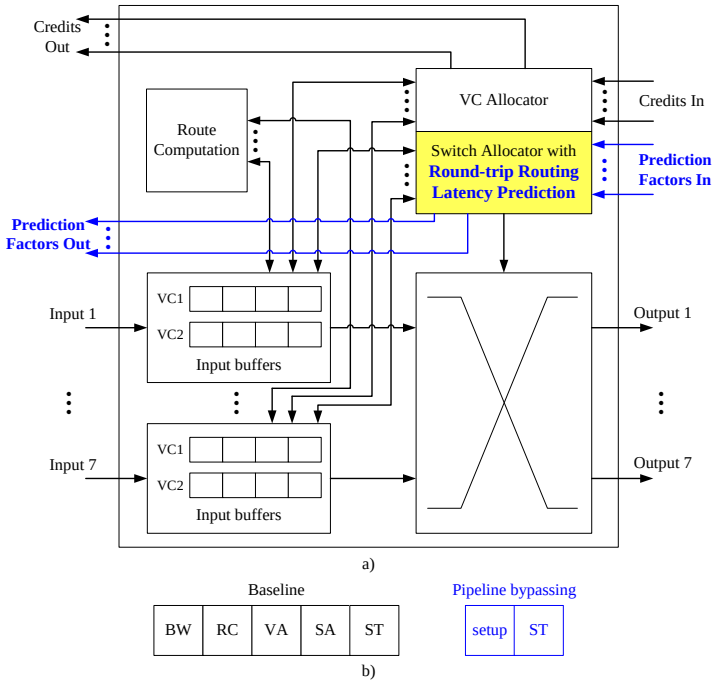

Fig. 3. a) A packet-switched credit-based virtual channel router supporting round-trip routing latency prediction; b) Pipeline stages in baseline and pipeline bypassing

Fig. 3 a) shows the microarchitecture of our router, which is a state-of-the-art packet-switched credit-based Virtual Channel (VC) [16] router. It is enhanced by supporting round-trip routing latency prediction implemented in the Switch Allocator (SA) module. Two virtual channels (VC1 for memory request packets and VC2 for memory response packets) are used to break deadlocks induced by the dependency of memory requests and memory responses. The router is designed as a typical microarchitecture with 5 logical stage pipeline[17], as shown in Fig. 3 b). A packet (its type can be "memory request" or "memory response"), upon arriving at an input port, is first written into the input buffer according to its input VC in the Buffer Write (BW) pipeline stage. In the next stage, the routing logic performs Route Computation (RC) to determine the output port for the packet. The packet then arbitrates for a VC corresponding to its output port in the VC Allocation (VA) stage. Upon successful allocation of a VC, the packet proceeds to the Switch Allocation (SA) stage where it arbitrates for the switch input and output ports. On winning the output port, the packet is then read from the input buffer and proceeds to the Switch Traversal (ST) stage, where it traverses the crossbar to be sent over the physical link finally. To accelerate the packet transmission speed over the router, a mechanism called "Pipeline bypassing"[17] is adopted. The BW, RC, VA and SA stages are combined and performed in the first stage which is named the "setup stage" where the crossbar is set up for packet traversal in the next cycle while simultaneously allocating a free VC corresponding to the desired output port. Therefore, moving one hop takes 1 clock cycle in our NoC. If there is a port conflict in the switch allocation between the packets in the two VCs, the packet with the higher priority is prioritized over the other. To support memory access equalization, we use the round-trip routing latencies of memory accesses as the prioritization base, since considering the outward trip and the return trip of an memory as a whole is more reasonable. The packet with longer round-trip routing latency has higher priority to go through the router. When the packet is traversing in the network, how much time will be taken in its future routing path is not known exactly but can be estimated. The next describes how to predict the round-

Valid	srcX	srcY	dstX	dstY	dstZ	dstZ	WT (Waiting Latency)	VC id	Type	Payload
								1	READ	Read Address
								1	WRITE	Write Address + Write Data
								2	RDATA	Read Data
								2	WACK	Write Acknowledgement

Fig. 4. Packet format

trip routing latency of a memory access.

(II) Predicting Round-trip Routing Latency

An entire memory access is a round-trip one containing two parts: memory request (read or write) in outward trip and memory response (read data or write acknowledgement) in return trip, so its appearance is a memory request packet in the first "memory request" phase of its transmission and a memory response packet in the second "memory response" phase of its transmission. The round-trip routing latency (notated as L) of a memory access can calculated by Formula (1).

$$L = (DL_p + WL_p) + (DL_f + WL_f) \tag{1}$$

The part in the first parentheses is the time that a memory access has consumed during its past transmission, which contains DL_p and WL_p representing the distance latency and the waiting latency[4] in the past transmission respectively. The part in the second parentheses is the time of the remaining transmission of a memory access, which includes DL_f and WL_f represents the distance latency and the waiting latency in the future transmission respectively. Because our network adopts the deterministic DOR X-Y-Z routing strategy, DL_f is deterministic. Therefore, Formula (1) is refined as:

$$L = DL_t + WL_p + WL_f \tag{2}$$

where DL_t is the total round-trip distance latency that is equal to DL_p plus DL_f.

To predict the round-trip routing latency, we need to calculate DL_t, WL_p, and WL_f.

(i) Calculating DL_t

DL_t is known in the deterministic routing network and can be calculated by Formula (3) according to the coordinates of the source and the destination.

$$DL_t = 2 \cdot (|X_{src} - X_{dst}| + |Y_{src} - Y_{dst}| + |Z_{src} - Z_{dst}|) \tag{3}$$

where X_{src},Y_{src}, and Z_{src} are the X, Y, and Z coordinates of the source, and X_{dst}, Y_{dst}, and Z_{dst} are the X, Y, and Z coordinates of the destination. They can be extracted from the packet (see Fig. 4).

(ii) Obtaining WL_p

WL_p is obtained from the "waiting latency (WL)" field in the packet. Fig. 4 illustrates the packet format. When a memory access starts, "WL" is initialized as zero in its memory request packet. The initial value of "WL" in its memory response packet is equal to the "WL" value of its memory request packet at the time when it reaches the destination shared memory. "WL" is incremented by 1 per clock cycle when the memory request packet or the memory response packet is blocked in the buffer due to the arbitration failure.

(iii) Estimating WL_f

[4]Distance latency is the transmission time of a packet without any contention, which is determined by the hop count and the clock cycles per hop. Waiting latency is the time consumed by a packet when it has to wait in the buffer due to its failure of winning the arbitration.

WL_f represents the possible waiting latency of a memory access in its future transmission. To estimate WL_f, we propose to use the number of the occupied items of the related input buffers in the downstream routers along the remaining routing path of a memory access. For instance, when a packet (notated as A) is going to its 1st downstream router (notated as R) through the inport whose input buffer has 2 packets (the first and the second are notated as $B1$ and $B2$ respectively), it will have to wait 1 hop (1 cycle in our design) for the departure of packet $B2$ until it passes through router R, since packet $B1$ leaves router R at the same time when packet A enters router R. Therefore, the future waiting time of packet A in its 1st downstream router R is considered to be 2. WL_f is estimated to be the sum of the count of the occupied items of the related input buffers in all downstream routers along the remaining routing path. Two steps are used to estimate WL_f as follows:

1) The list (notated as $\mathbb{R}$) of the downstream routers in the remaining routing path is obtained according to the packet's source and destination coordinates.
2) WL_f is calculated by Formula (4).

$$WL_f = \sum_{R \in \mathbb{R}} FWL(R) \tag{4}$$

where $FWL(R)$ is the function of calculating the future waiting latency in the downstream router R and shown in Formula (5).

$$FWL(R) = \begin{cases} \eta_R - \delta_R, & when \quad \eta_R > \delta_R \\ 0, & when \quad \eta_R \leqslant \delta_R \end{cases} \tag{5}$$

where δ_R is the hop count from the current router to the downstream router R and η_R is the number of the occupied items of the related input buffer in the downstream router R.

The two-step prediction method above considers the potential waiting latency due to the Head-of-Line (HoL) blocking induced by the packets that have existed in the input buffers in the remaining routing path. It does not predict the possible waiting latency due to the resource contention with other packets because the arrival time of other packets is undetermined and contention is hard to be described properly and estimated accurately.

(III) Implementing the prediction method

The two-step prediction method is generic and can be simplified when in concrete implementation. From Formula (5), we can see that the packet will not wait in the downstream router R if the number of packets in the related input buffer in the downstream router R is less than or equal to the hop count between the current router and the downstream router R. Therefore, the first step can be simplified to get the nearest 3 downstream routers in the remaining routing path since the depth of the input buffer in our router is only 4. Fig. 5 shows the pseudocode of estimating WL_f in our router. According to the X, Y, and Z coordinates of the current router and the destination of the packet, $DownstreamRouterExist()$ judges whether the downstream router exists, while $GetDownstreamRouter()$ gets the downstream router. In the router design, all of η_R in the nearest 3 routers come together to be the "Prediction Factors In" input and all of η_R in the current router form the "Prediction Factors Out" output, as shown in Fig. 3. η_R keeps track of the number of the occupied items of the related input buffer in the downstream router R. Its implementation is similar with the "Credit" signal. Each input buffer has a corresponding η_R counter, which increments when a new packet goes into the buffer and decrements when a packet departs the input buffer. η_R is the counter value.

```
WL_f = 0;
if DownstreamRouterExist(1) == true then
    R1 = GetDownstreamRouter(1);
    WL_f = WL_f + (η_R1 − 1);
end if //the 1st downstream router
if DownstreamRouterExist(2) == true then
    R2 = GetDownstreamRouter(2);
    WL_f = WL_f + (η_R2 − 2);
end if //the 2nd downstream router
if DownstreamRouterExist(3) == true then
    R3 = GetDownstreamRouter(3);
    WL_f = WL_f + (η_R3 − 3);
end if //the 3rd downstream router
```

Fig. 5. Pseudocode of estimating WL_f

D. Hardware Cost

The router design is synthesized in Synopsys® Design Compiler with TSMC® 45nm process. Table I lists the logic synthesis results excluding the wire cost. As stated in the previous paragraph, the extra hardware for implementing the prediction method is some η_R counters, so the extra hardware overhead is marginal. The extra area values are shown in the parentheses in Table I. The extra hardware does not degrade the frequency.

TABLE I
LOGIC SYNTHESIS RESULTS OF THE ROUTER

	Area	Frequency
Combinational Logic (RC+VA+SA+ST)	34029.30μm^2 (798.35μm^2) (36.17k NAND gates)	1.96 GHz (0.51 ns)
Sequential Logic (Input Buffers)	24545.19μm^2 (593.21μm^2) (26.09k NAND gates)	

Note: The area of a NAND gate with two inputs is 0.9408 μm^2.

IV. Experiments and Results

A. Experimental Setup

We implement a cycle-accurate homogenous many-core NoC simulator with Verilog, as shown in Fig. 1. The simulator models the processor cores, the shared memories and the NoC. The NoC has a 3D mesh topology and its size is configurable. The router is designed as described in Subsection III-C. To evaluate the performance of our proposal, uniform synthetic traffic patterns are considered. The random traffic represents the most generic case, where each processor core sends in-order memory accesses to the shared memories distributed in all nodes with a uniform probability. The target memories are selected randomly. In experiments, each processor core begins to generate 10,000 memory requests (read or write) to randomly selected destination shared memories when an experiment starts, and an experiment finishes after all processor cores receive their related 10,000 memory responses (read data or write acknowledgement). All of the experiments are performed with a variety of network sizes and packet injection rates. For performance comparison, we take the classic widely used round-robin arbitration as the counterpart and evaluate maximum latency (ML), average latency (AL), and latency standard deviation (LSD) that are defined by Formula (6), (7), and (8) respectively:

$$ML = \max(\{L_1, L_2, \cdots, L_N\}) \tag{6}$$

$$AL = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{7}$$

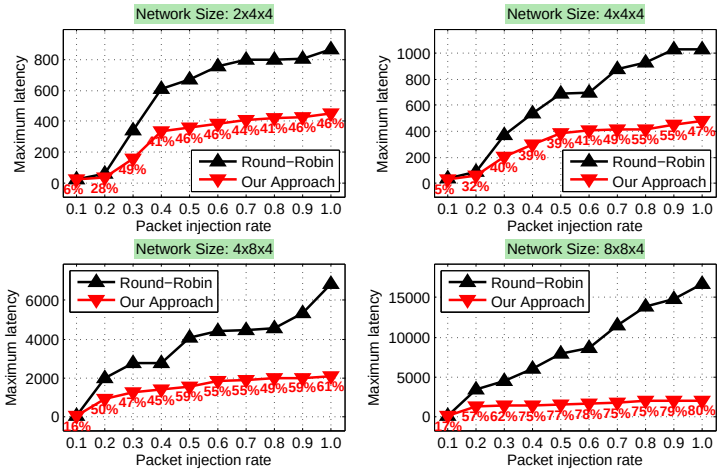

Fig. 6. Comparison of maximum latency between round-robin arbitration and our approach under varied network sizes and packet injection rates

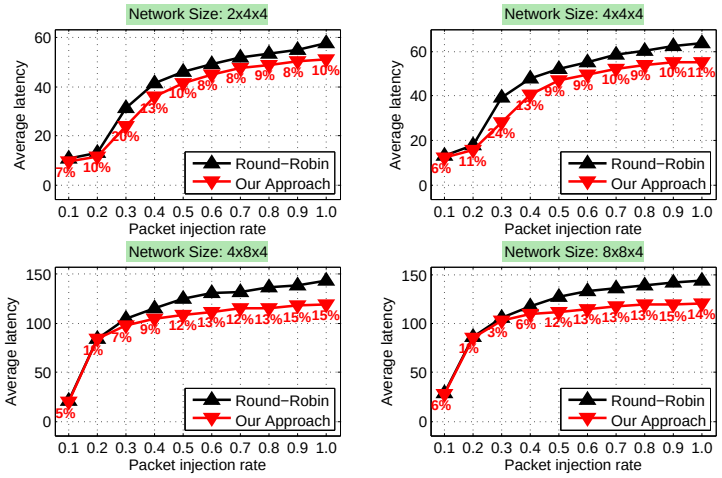

Fig. 7. Comparison of average latency between round-robin arbitration and our approach under varied network sizes and packet injection rates

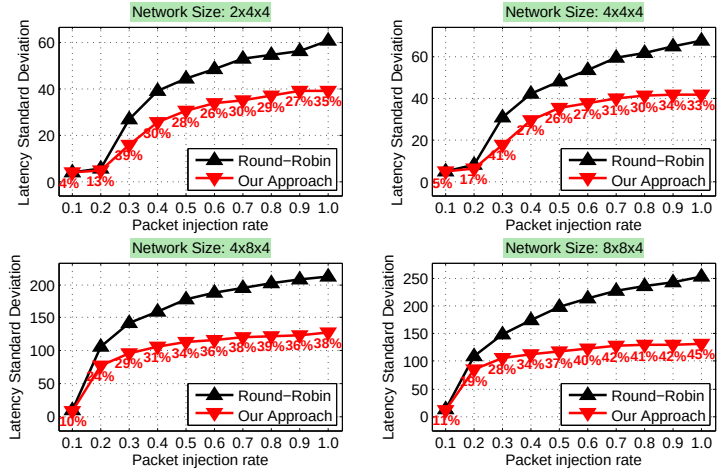

Fig. 8. Comparison of latency standard deviation between round-robin arbitration and our approach under varied network sizes and packet injection rates

$$LSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(L_i - AL)^2} \tag{8}$$

where N is the total number of memory accesses and L_i is the round-trip routing latency of the memory access with the id of i.

B. Performance Evaluation

Fig. 6, 7, and 8 respectively plot maximum latency, average latency, and LSD under different network sizes and packet injection rates. The percentages in the figures indicate the performance improvement of our approach in comparison to the round-robin arbitration. From these figures, we can see that:

- Compared with the classic round-robin arbitration, our approach can has lower maximum latency, average latency, and LSD, thus making the latencies of memory accesses more balanced and achieving the goal of memory access equalization.
- As the network size is scaled up and the packet injection rates increases, our approach can basically gain more performance improvement in terms of maximum latency, average latency and LSD, meaning that, under large-scale network size with a large number of memory accesses, the classic round-robin arbitration has large latency gap of different memory accesses and our approach can balance the latencies of memory accesses well. For instance, under the network size of $8\times8\times4$ and the packet injection rate of 1.0, in comparison to the round-robin arbitration, the maximum latency, the average latency and the LSD are improved by 80%, 14%, and 45% respectively, which are the maximum performance improvement we obtain in the experiments.

To further evaluate the memory access equalization, we collect the number of memory accesses with different round-trip routing latencies. Fig. 9 exhibits the latency dispersion of memory accesses. From the figure, we can see that:

- Our approach reduces the number of memory accesses with high latencies in comparison to the classic round-robin arbitration.
- Because the total number of memory accesses are the same in both our approach and the classic round-robin arbitration, our approach increases the number of memory accesses with low latencies.

Therefore, our approach makes the latency dispersion curve be narrower and higher so that the memory access equalization is achieved.

V. Concluding Remark

In 3D many-core NoCs, as the network size is scaled up and the number of memory accesses increases largely, the performance (latency) gap of different memory accesses becomes bigger. Some memory accesses with very high latencies exist, thus negatively affecting the system performance. To achieve the goal of memory access equalization (i.e. balancing the latencies of memory accesses and reducing the number of memory accesses with high latencies), the paper proposes to prioritize the memory access packets through predicting their round-trip routing latencies. The communication distance and the number of the occupied items in the input buffers in the remaining routing path are used to predict the future possible waiting time of a memory access on-the-fly, which is part of the round-trip routing latency. Experiments with varied network sizes and packet injection rates demonstrate that our approach outperforms the classic round-robin arbitration in terms of maximum latency, average latency, and LSD and can achieve the goal of memory access equalization. In the future, we plan to apply real application workloads to evaluate the performance of our proposal, and link and optimize our approach on other irregular or heterogeneous 3D many-core NoCs.

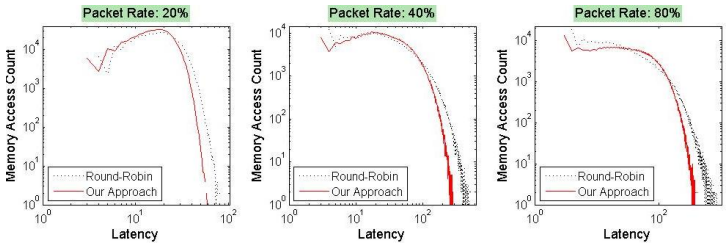

Fig. 9. Comparison of memory access latency dispersion between round-robin arbitration and our approach under varied packet injection rates in $4\times4\times4$ network.

Acknowledgment

The research is partially supported by the Hunan Natural Science Foundation of China (No. 2015JJ3017), and the Doctoral Program of the Ministry of Education in China (No. 20134307120034), and the National Natural Science Foundation of China (No. 61402500).

References

[1] J. D. Owens, W. J. Dally *et al.*, "Research challenges for on-chip interconnection networks," *IEEE MICRO*, vol. 27, no. 5, pp. 96–108, Oct. 2007.

[2] D. H. Kim, K.Athikulwongse, H. M. *et al.*, "3d-maps: 3d massively parallel processor with stacked memory," in *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, Feb. 2012, pp. 188–189.

[3] D. Fick, R. Dreslinski, B. Giridhar *et al.*, "Centip3de: A 3930 dmips/w configurable near-threshold 3d stacked system with 64 arm cortex-m3 cores," in *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, Feb. 2012, pp. 190–191.

[4] R. Dreslinski, D. Fick, B. Giridhar *et al.*, "Centip3de: A 64-core, 3d stacked near-threshold system," *IEEE Micro*, vol. 33, no. 2, pp. 8–16, Feb. 2013.

[5] M. Wordeman, J. Silberman, G. Maier *et al.*, "A 3d system prototype of an edram cache stacked over processor-like logic using through-silicon vias," in *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, Feb. 2012, pp. 186–187.

[6] Y. Zhang, L. Li, Z. Lu *et al.*, "Performance and network power evaluation of tightly mixed sram nuca for 3d multi-core network on chips," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, Jun. 2014, pp. 1961–1964.

[7] D. Genius, "Measuring memory access latency for software objects in a numa system-on-chip architecture," in *Proceedings of the 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, Jul. 2013, pp. 1–8.

[8] Z. Majo and T. R. Gross, "Memory system performance in a numa multicore multiprocessor," in *Proceedings of the 4th Annual International Conference on Systems and Storage*, May 2011, pp. 1–10.

[9] S. Rixner, W. Dally, U. Kapasi, P. Mattson, and J. Owens, "Memory access scheduling," in *Proceedings of the 27th International Symposium on Computer Architecture (ISCA)*, Feb. 2000, pp. 128–138.

[10] O. Mutlu and T. Moscibroda, "Stall-time fair memory access scheduling for chip multiprocessor," in *Proc. of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2007, pp. 146–160.

[11] M. Daneshtalab, M. Ebrahimi, J. Plosila, and H. Tenhunen, "Cars: Congestion-aware request scheduler for network interfaces in noc-based manycore systems," in *Proc. of the Design, Automation and Test in Europe Conf. (DATE'13)*, Mar. 2013, pp. 1048–1051.

[12] D. Kim, S. Yoo, and S. Lee, "A network congestion-aware memory controller," in *Proc. of the 4th ACM/IEEE International Symposium on Networks-on-Chip*, May 2010, pp. 257–264.

[13] W. Jang and D. Pan, "An sdram-aware router for networks-on-chip," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 10, pp. 1572–1585, Oct. 2010.

[14] T. Pimpalkhute and S. Pasricha, "Noc scheduling for improved application-aware and memory-aware transfers in multi-core systems," in *Proc. of the 27th International Conference on VLSI Design and the 13th International Conference on Embedded Systems*, Jan. 2014, pp. 234–239.

[15] A. Sharifi, E. Kultursay, M. Kandemir, and C. Das, "Addressing end-to-end memory access latency in noc-based multicores," in *Proc. of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2012, pp. 294–304.

[16] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, Feb. 1992.

[17] A. Kumar, L. Peh, P. Kundu, and N. Jha, "Express virtual channels: Towards the ideal interconnection fabric," in *Proc. of the 34th Annual Int'l Symp. on Computer Architecture (ISCA)*, Dec. 2007, pp. 150–161.