Appears in Proceedings of the 19th International Symposium on Computer Architecture & Digital Systems (CADS)[*]

NOC Characteristics of Cloud Applications

Pejman Lotfi-Kamran[*], Mehdi Modarressi[†], and Hamid Sarbazi-Azad[‡*]
[*]School of Computer Science, Institute for Research in Fundamental Sciences (IPM)
[†]School of Electrical and Computer Engineering, University of Tehran
[‡]Department of Computer Engineering, Sharif University of Technology

***Abstract*—Cloud applications have abundant request-level parallelism, and as a result, many-core server processors are good candidates for their execution. A key component in a many-core processor is the network-on-chip (NOC) that connects cores to cache banks and memory, and acts as the medium for delivering instructions and data to the cores. While cloud applications are an important class of massively-parallel workloads that benefit from many-core processors and networks-on-chip, there is no comprehensive study for the NOC requirements of these workloads. In this work, we use full-system simulation and a set of cloud applications to study the characteristics and requirements of these applications with respect to networks-on-chip. We find that NOC latency is the most important optimization criterion for these workloads. As NOC traffic of these workloads is relatively low and approximately follows uniform traffic, we find that knobs like routing algorithm and buffer size that mostly affect NOC bandwidth, beyond a certain point, have little impact on the performance of these workloads. On the other hand, techniques that reduce NOC latency directly improve the performance of cloud applications.**

I. Introduction

Major online service providers employ large networks of datacenters to offer a growing number of services, such as Web search, social networking, and media streaming. These services handle independent requests that do not share any state. Due to serving independent requests, cloud workloads are inherently parallel and best suited to be run on many-core processors [1].

One of the key components in a many-core processor that affects the performance of cloud workloads is the on-chip network [2], [3], [4]. Due to the importance of cloud computing, it is necessary to design processors that run cloud workloads efficiently. Unfortunately, there is no comprehensive study for the network-on-chip (NOC) requirements of cloud workloads.

To have an efficient network-on-chip architecture, it is critical to have a detailed understanding of the communication behavior of the target applications. Without this knowledge, designers can hardly fit the NOC to the on-chip traffic characteristics; hence NOC may either come with more resources than needed, i.e., overprovisioned, or fall short of the bandwidth/latency requirement. While the former will reduce resource utilization, the latter may lead to considerable performance degradation, as a NOC is an important component that influences overall system performance.

Several previous studies have analyzed the traffic behavior of GPU workloads [5], PARSEC suite [6], and scientific and signal processing applications [7]. In this paper, we focus on the emerging cloud workloads, analyze their traffic behavior on NOC-based multi-core processors, and shed light on suitable NOC configurations that better fit the behavior.

We use full-system simulation and a number of cloud applications to study NOC requirements and characteristics of cloud applications. We find that due to good L1 performance, NOC traffic of cloud applications is relatively low. Moreover, the traffic pattern follows uniform distribution with very few source-destination pairs that generate higher than average traffic. As a significant fraction of NOC traffic in cloud applications is due to L1 instruction misses, cloud applications are sensitive to the NOC latency. However, as the traffic in the network is low (in a typical setting), NOC bandwidth is not a constraint of performance under realistic settings.

We find that a simple dimensions-order routing (DOR) algorithm performs as good as sophisticated adaptive routing algorithms (e.g., [8], [9], [10], [11]). Moreover, our analysis shows that cloud applications significantly benefit from latency reduction techniques, such as those that target reducing network diameter [12], [13], shortening the router pipeline stages [14], [15], [16], [17], [18] or facilitate single-cycle multi-hop traversal [19], [20].

II. Background

In this section, we review some of the main characteristics of cloud applications and networks-on-chip.

A. Cloud Applications

Recent research [1] indicates that cloud workloads, as a class, have characteristics that distinguish them from desktop, scientific, and traditional server workloads. These workloads, e.g., Web search or media streaming, are inherently parallel as they serve a large number of requests that are overwhelmingly independent. Essentially request independence, and the parallelism that is derived from it, makes many-core processors attractive for execution of cloud applications.

Another characteristic of cloud applications is their large instruction footprint [1]. Cloud applications typically have complex control flow and multi-megabyte instruction footprints. Due to the large footprint, instructions do not fit in L1 instruction caches, and as such, cores executing cloud applications encounter frequent instruction misses. For each miss, be an instruction or a data miss, a request needs to be sent to the last-level cache (LLC) and the response needs to be delivered to the core.

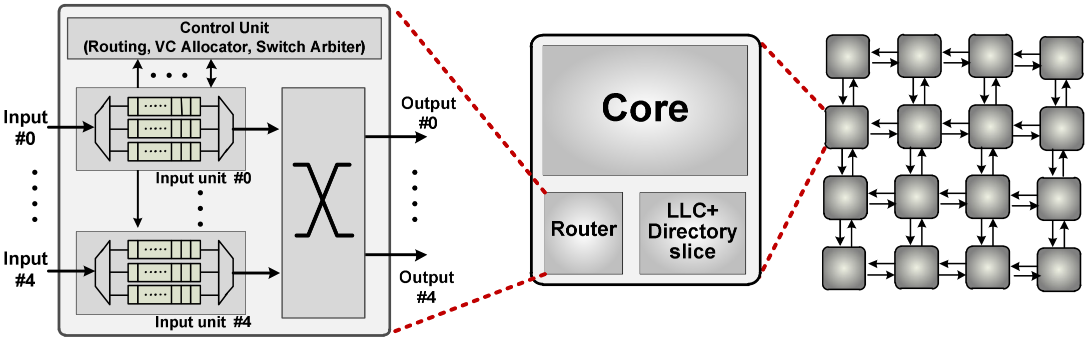

Fig. 1. Elements of tiled many-core processors.

B. Mesh On-Chip Network

Many-core processors are suitable for execution of cloud applications. These processors usually employ a tiled organization with a fully distributed last-level cache (LLC) [21]. For the purpose of connecting private caches of the cores to the LLC and memory channels, a mesh network-on-chip is usually used. A 16-core tiled-based processor is shown in Figure 1. Each tile consists of a core, a slice of the distributed last-level cache, a directory slice, and a router. The tiles are linked via a packet-based, multi-hop interconnect in a mesh topology.

As networks-on-chip have significant impact on the performance of many-core processors, they need to be designed and tuned carefully. There are many parameters that influence how a network-on-chip operates. These parameters include routing algorithm, size of the buffers, channel width, and the delay of each router. In this work, we study the impact of each parameter on the performance of many-core processors when cloud applications are being executed.

III. Experimental Methodology

Table I summarizes key elements of our experimental setup, with the following sections detailing the evaluated designs, workloads, and simulation infrastructure.

A. Processor Parameters

We model a processor with 16 cores, 8 MB of last-level cache, and four DDR3-1600 memory channels. Core microarchitecture is modeled after an ARM Cortex-A15, a 3-way out-of-order design with 32 KB L1-I and L1-D caches. Cache line size is 64 bytes.

Our baseline is a mesh-based tiled processor, as shown in Figure 1. The 16 tiles are organized as a 4-by-4 grid, with each tile containing a core, a slice of the LLC, and a directory node. A hop in the baseline mesh consists of two-stage router pipeline followed by a single-cycle link traversal for a total of three cycles per hop at zero load. The router performs routing, VC allocation, and speculative crossbar (XB) allocation in the first cycle, followed by XB traversal in the second cycle, and link traversal in the next cycle. Each router port in the baseline mesh has three VCs to guarantee deadlock freedom across three message classes: requests, snoops, and responses (each message class has just one VC). Each VC is five flits deep. We vary various parameters of the baseline mesh according to Table I to study the sensitivity of the performance of cloud applications to these parameters.

B. Workloads

We use cloud workloads from CloudSuite [23]. The workloads include Data Serving, MapReduce, Media Streaming, SAT Solver, Web Frontend, and Web Search. Two of the workloads – SAT Solver and MapReduce – are batch, while the rest are latency-sensitive and are tuned to meet the response time objectives. Prior work [1] has shown that these workloads have characteristics representative of the broad class of cloud workloads.

C. Simulation Infrastructure

We measure the performance of various processor designs using Flexus full-system simulation [24]. Flexus extends the Virtutech Simics functional simulator with timing models of cores, caches, on-chip protocol controllers, and interconnect. Flexus models the SPARC v9 ISA and is able to run unmodified operating systems and applications. Flexus uses BookSim 2.0 network simulator [25] for modeling the on-chip network.

We use the SimFlex multiprocessor sampling methodology [24]. Our samples are drawn over an interval of 10 seconds (30 seconds for Media Streaming) of simulated time. For each measurement, we launch simulations from checkpoints with warmed caches and branch predictors, and run 100 K cycles to achieve a steady state of detailed cycle-accurate simulation before collecting measurements for the subsequent 50 K cycles. We use the ratio of the number of application instructions to the total number of cycles (including the cycles spent executing operating system code) to measure performance; this metric has been shown to accurately reflect overall system throughput of multiprocessors [24]. Performance measurements are computed with 95% confidence and an error of less than 4%.

IV. Evaluation

We first examine the NOC traffic behavior of cloud applications. We then present sensitivity of cloud applications'

TABLE I
EVALUATION PARAMETERS.

Parameter	Baseline	Variations
Technology	32 nm, 2 GHz	—
Processor features	16 cores, 8 MB NUCA LLC, Four DDR3-1600 memory channels	—
Core	ARM Cortex-A15-like: 3-way out-of-order, 64-entry ROB 16-entry LSQ	—
Topology	4×4 2D Mesh	—
Routing	DOR [22]	Local adaptive [22]; RCA [9]
Per-hop latency	3	2; 1
Virtual channels/Message class	1	—
Flit buffers/VC	5	1; 3; 7; 9
Channel width	128 bits	32 bits; 64 bits; 256 bits
Workload	CloudSuite [23]	—

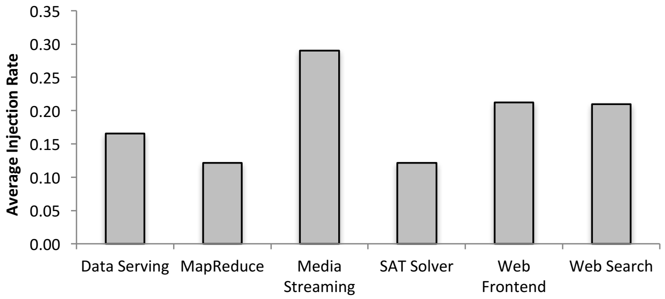

Fig. 2. The average network injection rate (flit/node/cycle) of cloud applications.

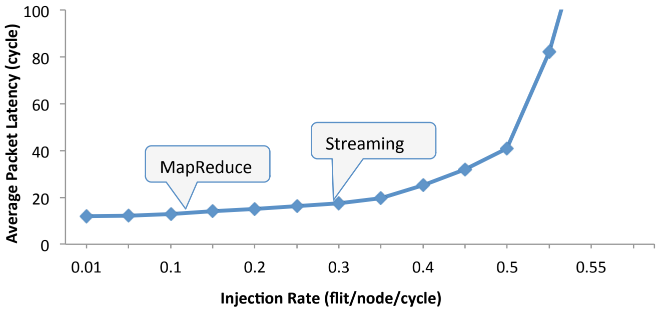

Fig. 3. Packet injection rate of two cloud applications with the highest and lowest injection rates within the load-latency graph.

performance to various NOC parameters. Our baseline is a mesh network with one virtual channel per message class, five buffers per VC, 128-bit channels, and 3-cycle routers with dimension-order routing (DOR). All performance numbers are normalized to the baseline.

A. Communication rate and behavior

Figure 2 shows the average injection rate of the considered benchmarks. As the figure demonstrates, the traffic injection rate varies from 0.12 flit/node/cycle in MapReduce to 0.29 flit/node/cycle in Media Streaming. Media Streaming is a data-intensive application that deals with large online video data: it processes and transmits data at very high rates; hence the on-chip traffic is higher in Media Streaming than the rest of the workloads. Even for Media Streaming, the NOC working condition is well below the saturation point.

To investigate this issue, we show where the traffic rates of the CloudSuite benchmarks lie in the range between the zero-load and the saturation point. To this end, a synthetic traffic is generated based on the spatial traffic pattern of the CloudSuite applications (uniform with a few hotspot nodes). Figure 3 shows the load-latency graph of a 4×4 mesh NOC under this traffic. The results confirm that the injection range of CloudSuite applications falls into the middle of the traffic range: it is higher than the zero-load traffic, but is well below the saturation point.

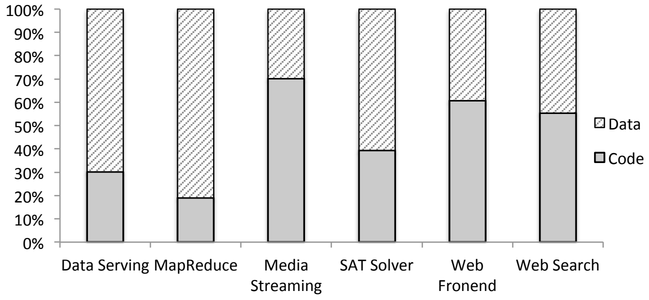

Fig. 4. Contribution of Instruction and Data misses to the on-chip traffic.

As cloud applications have large instruction footprints, we investigate the contribution of data and instruction misses to the network traffic. Figure 4 shows the contribution of instruction and data misses to the network traffic. We classify a packet as *code* if it is, directly or indirectly, injected into the network because of an L1-I cache miss. Otherwise, the packet is classified as *data*. In every CloudSuite application, a significant fraction of on-chip traffic is due to instruction misses. The contribution of instruction misses to the on-chip traffic ranges from 19% in MapReduce to 70% in Media Streaming. As processors are particularly sensitive to the miss penalty of instruction caches, network latency is one of the

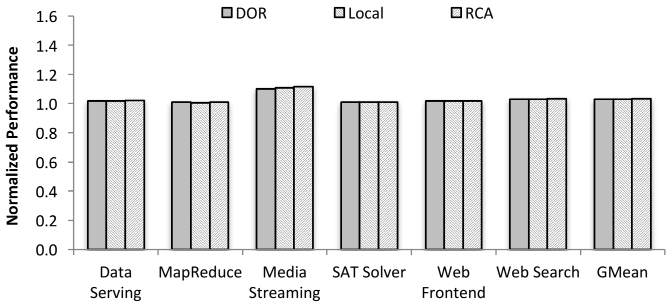

Fig. 5. System performance for various routing algorithms, normalized to the baseline mesh. The baseline uses dimension-order routing (DOR) with 1 VC/message class. The other routing algorithms include local adaptive (Local) [22] and RCA [9]. All systems have 2 VCs/message class. To guarantee deadlock avoidance, systems with adaptive routing algorithms use one VC as Escape Channel.

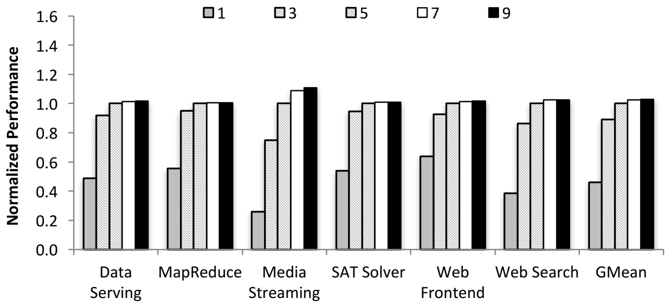

Fig. 6. System performance for various buffer sizes per virtual channel, normalized to the baseline mesh. The baseline has one VC with 5-flit buffers.

primary optimization criterion for cloud applications.

B. Sensitivity to the Choice of Routing Algorithm

One of the components in a router that influences the flow of packets in the network is the routing algorithm. For every packet and in every hop, the routing algorithm determines to which output port the packet should be sent to for the packet to be delivered to the destination. A common class of routing algorithms, called deterministic, decides the output port based on a pre-defined algorithm and regardless of the NOC congestion status. For example, dimension-order routing (DOR) algorithm directs packets solely based on the destination. Alternatively, adaptive routing algorithms take the status of the network into consideration to send the packet on a path with minimum congestion. Adaptive routing algorithms are further classified to (1) *local* if they rely on information within a router for the purpose of choosing the output port and (2) *regional (or global)* otherwise. Adaptive routing algorithms are believed to play a key role in reducing the blocking delay that packets experience.

To study the effectiveness of routing algorithms on the performance of cloud applications, we measure the effect of three routing algorithms, DOR [22] (deterministic), Local adaptive [22], and RCA [9] (i.e., regional adaptive) on improving the performance of cloud applications. All of the parameters of the processor are similar to the baseline processor with the only exception of the routing algorithm and the number of virtual channels (VCs). As adaptive routing algorithms require a deadlock avoidance mechanism [22], in all cases, each message class has two VCs and the extra VC is used as Escape Channel for deadlock avoidance [26].

Figure 5 shows the results of this experiment. All performance numbers are normalized to the baseline. Except on Media Streaming, which has the highest traffic load and deviation from the uniform spatial load distribution, adaptive routing algorithms have no effect on the performance of cloud applications. For Media Streaming, Local and RCA routing algorithms improve system performance over DOR by 11% and 12%, respectively. Across all applications, the performance improvement of Local and RCA over DOR is 3% and 3%, respectively. Another observation is that local and regional routing algorithms have the same effect on the performance of cloud applications. Even on Media Streaming, which is the most sensitive studied application to the choice of routing algorithm, Local and RCA yield almost the same performance. There are two reasons for cloud applications not being sensitive to the choice of routing algorithm: (1) little traffic in the network as shown in Figure 2 and (2) traffic pattern approximately follows uniform distribution, and DOR works best for uniform traffic [9]. These results suggest that a simple DOR algorithm is the right choice for cloud applications to balance simplicity and performance.

C. Sensitivity to Buffer Size

Another component in a router that influences the way that packets travel through the network is the buffer. Usually buffers are placed at the input ports of a router: when a packet's flit passes a link and goes from Router A to Router B, it gets buffered in the input port of Router B. A packet may get blocked in a router (e.g., the output port is busy sending another packet). In such a case, if the input port has free buffers, subsequent flits (either flits of the blocked packet or other packets) can pass the link and go to the input buffer, despite the fact that previous flit is stuck in the input buffer. Because buffers impact the performance of a network-on-chip, one of the important design choices of a NOC is the size of the buffer.

In this part, we study the impact of buffer size on the performance of cloud applications. Figure 6 shows the performance of a processor (similar to the baseline) when the size of the buffer changes from 1 to 9. All performance numbers are normalized to the performance of the baseline processor that has 5-flit buffer. When the size of the buffer is smaller than the packet length, if a packet is blocked, it occupies multiple channels, and does not let other packets use the channels (or virtual channel if a channel has more than one virtual channel). Consequently, the utilization of the network decreases, which leads to poor performance. In our case, the maximum size of a packet is 5 (64B cache

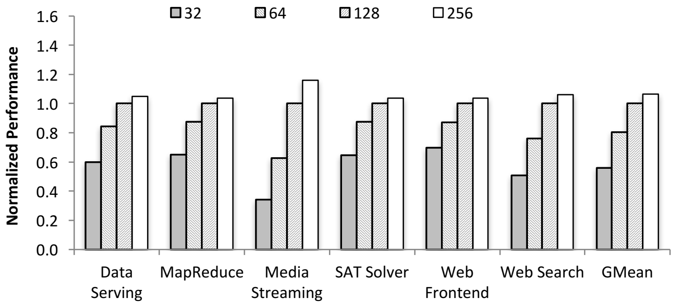

Fig. 7. System performance for various channel widths, normalized to the baseline mesh. The baseline has 128-bit channels.

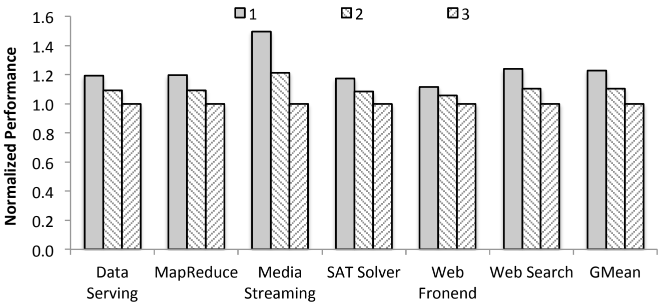

Fig. 8. System performance for various router delays, normalized to the baseline mesh. The baseline has routers with the delay of 3 cycles.

blocks and 128-bit channels), and as such, 1- and 3-entry buffers are not enough to avoid channel occupation when a packet is blocked. Figure 6 clearly shows that 1- and 3-entry buffers significantly hurt performance of cloud applications (11% and 55% lower performance as compared to 5-entry buffers, respectively). Moreover, as the traffic in the network is low, when the buffer size becomes equal to the maximum packet size (i.e., 5), the system performance levels off. Beyond five, increasing the size of the buffer has little positive impact on the performance of cloud applications. This experiment suggests 5-entry buffers are the right choice for the network-on-chip of the baseline processor. A conservative design may suggest over-provisioning the buffering capacity to eliminate any possible buffer-induced performance loss. However, fitting the buffer size to traffic behavior brings about more energy- and area-efficiency, as it has been shown that buffers are the highest contributor to the total NOC power consumption and area [27].

D. Sensitivity to Channel Width

Another parameter of a network-on-chip that may influence the performance is channel width. Usually size of a packet is larger than channel width. When a large packet wants to pass a channel, it needs to be broken into segments, called flits, where size of a flit is equal to the size of a channel. The wider the channel, the lower the number of flits per packet and the lower the overhead of serialization. On the other hand, the wider a channel is, the higher the area overhead becomes. Choosing the size of the channels is one of the key design choices of a network-on-chip.

In this section, we study the effect of channel width on the performance of cloud applications. Figure 7 shows the performance sensitivity of a 16-core processor running cloud applications to the channel width of its network-on-chip. All parameters of the processor are identical to the baseline with potentially one exception: channel width. All performance numbers are normalized to the baseline (baseline processor has 128-bit links in its channels). Figure 7 shows that 32-bit and 64-bit channels significantly hurt performance, as compared to the baseline 128-bit channels (20% and 45%, respectively). A data packet consists of 17 and 9 flits in 32- and 64-bit channels, respectively, while the same packet just consists of 5 flits in a 128-bit baseline network. The high serialization overhead associated with narrow channels makes them ineffective for cloud applications. On the other hand, increasing the size of the channel from 128 to 256 bits results in just 6% improvement in performance of the baseline processor. These data suggest having 128-bit channels in the network-on-chip of the baseline processor, as narrower channels significantly hurt performance and wider channels do not yield commensurate increase in performance.

E. Sensitivity to Per-Hop Latency

One of the parameters of a network-on-chip that may affect the performance is per-hop latency. A packet, on the way to the destination, should pass multiple hops (routers). The minimum delay that a packet experiences in each hop depends on the network-on-chip. While a packet may experience a delay longer than the minimum in a hop due to conflicts with other packets, the minimum per-hop delay is influential in determining the performance. In this part, we examine the sensitivity of cloud applications' performance on the per-hop latency in the network-on-chip.

Figure 8 shows the performance of cloud applications for networks with 1-, 2-, and 3-cycle delay per hop. All the parameters of the processor that is running cloud applications are identical to the baseline except for the per-hop latency. All performance numbers are normalized to the baseline. As expected, a decrease in per-hop latency translates into an increase in system performance. While all of the applications are sensitive to per-hop latency, Media Streaming shows the highest sensitivity. For Media Streaming, reducing the per-hop latency to 2 and 1 results in 21% and 50% higher system performance, respectively. Across all applications, networks with 2- and 1-cycle per-hop latency offer 11% and 23% higher performance, as compared to 3-cycle per-hop networks.

These results clearly indicate high sensitivity of cloud applications' performance to the network latency. One reason for such a high sensitivity of cloud applications to network latency is the large instruction footprints of these applications [1]. As a result of having large instruction footprints, a significant fraction of network traffic is due to instructions, as shown in Figure 4. While OoO cores, like the one used for the experiments, may overlap a data miss with some useful work, they become idle as a result of an instruction miss. This makes cloud applications extremely sensitive to L1 instruction miss penalty, which NOC is a major contributor to [28]. Consequently, research on reducing network latency is important for getting high performance on cloud applications.

V. Conclusions

Cloud applications, due to massive request-level parallelism, benefit from execution on many-core processors. A key component in a many-core processor is the network-on-chip. Using a set of cloud applications and a cycle-accurate simulator, this work studied the NOC characteristics of cloud applications and the requirements that they place on networks-on-chip. Our experiments showed that performance of cloud applications running on many-core processors depends heavily on the characteristics of the network-on-chip. We found that the majority of on-chip traffic in cloud applications is due to instruction misses. While the NOC bandwidth demand is moderate, due to sensitivity of cloud applications' performance on instruction misses' service time, these workloads are particularly sensitive to NOC latency. Our findings showed that knobs like the choice of routing algorithm and buffer size that are particularly useful for applications with high bandwidth demands are less effective in increasing performance of cloud applications. On the other hand, techniques that reduce NOC latency can significantly affect the performance of cloud applications.

References

[1] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware," in *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Mar. 2012, pp. 37–48.

[2] P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi, "Scale-Out Processors," in *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2012, pp. 500–511.

[3] P. Lotfi-Kamran, B. Grot, and B. Falsafi, "NOC-Out: Microarchitecting a Scale-Out Processor," in *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2012, pp. 177–187.

[4] B. Falsafi, B. Grot, and P. Lotfi-Kamran, "Network-on-chip using Request and Reply Trees for Low-latency Processor-memory Communication," Jul. 2017, US Patent 9,703,707.

[5] N. Goswami, R. Shankar, M. Joshi, and T. Li, "Exploring GPGPU Workloads: Characterization Methodology, Analysis and Microarchitecture Evaluation Implications," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, Dec. 2010, pp. 1–10.

[6] N. Barrow-Williams, C. Fensch, and S. Moore, "A Communication Characterisation of Splash-2 and Parsec," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, Oct. 2009, pp. 86–97.

[7] Z. Wang, W. Liu, J. Xu, B. Li, R. Iyer, R. Illikkal, X. Wu, W. H. Mow, and W. Ye, "A Case Study on the Communication and Computation Behaviors of Real Applications in NoC-Based MPSoCs," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, Jul. 2014, pp. 480–485.

[8] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, and Z. Navabi, "BARP-A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, Mar. 2008, pp. 1408–1413.

[9] P. Gratz, B. Grot, and S. W. Keckler, "Regional Congestion Awareness for Load Balance in Networks-on-Chip," in *Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2008, pp. 203–214.

[10] P. Lotfi-Kamran, A.-M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, "EDXY - A Low Cost Congestion-Aware Routing Algorithm for Network-on-Chips," *Journal of Systems Architecture*, vol. 56, no. 7, pp. 256–264, Jul. 2010.

[11] P. Lotfi-Kamran, "Per-Packet Global Congestion Estimation for Fast Packet Delivery in Networks-on-Chip," *The Journal of Supercomputing*, vol. 71, no. 9, pp. 3419–3439, Sep. 2015.

[12] H. Yang, J. Tripathi, N. E. Jerger, and D. Gibson, "Dodec: Random-Link, Low-Radix On-Chip Networks," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2014, pp. 496–508.

[13] U. Y. Ogras and R. Marculescu, ""It's a Small World After All": Noc Performance Optimization via Long-range Link Insertion," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 693–706, Jul. 2006.

[14] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA)*, Jan. 2001, pp. 255–266.

[15] R. Mullins, A. West, and S. Moore, "Low-Latency Virtual-Channel Routers for On-Chip Networks," in *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA)*, Nov. 2004, pp. 188–197.

[16] S. H. Seyyedaghaei Rezaei, A. Mazloumi, M. Modarressi, and P. Lotfi-Kamran, "Dynamic Resource Sharing for High-Performance 3-D Networks-on-Chip," *IEEE Computer Architecture Letters*, vol. 15, no. 1, pp. 5–8, Jan. 2016.

[17] B. K. Daya, C.-H. O. Chen, S. Subramanian, W.-C. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L.-S. Peh, "SCORPIO: A 36-core Research Chip Demonstrating Snoopy Coherence on a Scalable Mesh NoC with In-network Ordering," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA)*, Nov. 2014, pp. 25–36.

[18] P. Lotfi-Kamran, M. Modarressi, and H. Sarbazi-Azad, "An Efficient Hybrid-Switched Network-on-Chip for Chip Multiprocessors," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1656–1662, May 2016.

[19] T. Krishna, C.-H. O. Chen, W. C. Kwon, and L.-S. Peh, "Breaking the On-chip Latency Barrier Using SMART," in *Proceedings of the 19th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2013, pp. 378–389.

[20] P. Lotfi-Kamran, M. Modarressi, and H. Sarbazi-Azad, "Near-Ideal Networks-on-Chip for Servers," in *Proceedings of the IEEE 23rd International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2017, pp. 277–288.

[21] "Tilera TILE-Gx." [Online]. Available: http://www.mellanox.com/related-docs/prod_multi_core/PB_TILE-Gx36.pdf

[22] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, 1st ed. Morgan Kaufmann Publishers Inc., 2003.

[23] CloudSuite, http://cloudsuite.ch.

[24] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe, "SimFlex: Statistical Sampling of Computer System Simulation," *IEEE Micro*, vol. 26, no. 4, pp. 18–31, July-August 2006.

[25] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2013, pp. 86–96.

[26] L. M. Ni and P. K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Computer*, vol. 26, no. 2, pp. 62–76, Feb. 1993.

[27] T. Moscibroda and O. Mutlu, "A Case for Bufferless Routing in On-Chip Networks," in *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2009, pp. 196–207.

[28] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Reactive NUCA: Near-Optimal Block Placement and Replication in Distributed Caches," in *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2009, pp. 184–195.