On-Chip Network Evaluation Framework

Hanjoon Kim, Seulki Heo, Junghoon Lee, Jaehyuk Huh, John Kim
KAIST
Department of Computer Science
Daejeon, Korea
{hanj, sapiencex, ijij41, jhhuh, jjk12}@kaist.ac.kr

***Abstract*—With the number of cores on a chip continuing to increase, proper evaluation of on-chip network is critical for not only network performance but also overall system performance. In this paper, we show how a network-only simulation can be limited as it does not provide an accurate representation of system performance. We evaluate traditionally used open-loop simulations and compare them to closed-loop simulations. Although they use different methodologies, measurements, and metrics, we identify how they can provide very similar results. However, we show how the results of closed-loop simulations do not correlate well with execution-driven simulations. We then add simple extensions to the closed-loop simulation to model the impact of the processor and the memory system and show how the correlation with execution-driven simulations can be improved. The proposed framework/methodology provides a fast simulation time while providing better insights into the impact of network parameters on overall system performance.**

I. Introduction

With the number of cores on a chip continuing to increase, the on-chip network or network-on-chip (NoC) is becoming a critical part of future chip multiprocessor (CMP) systems, as the performance of on-chip networks impacts the overall system performance. Different design parameters of NoC such as network topology, routing algorithm, and router parameters affect not only network performance but also overall system performance. Thus, proper evaluation of NoC is critical to understand the impact of NoC.

Historically, interconnection networks have been evaluated using a synthetic traffic workload via *open-loop* simulations [8] – i.e., the network does not impact the traffic pattern injected into the network. This evaluation methodology can accurately evaluate *network* performance under different design parameters. However, this does not necessarily measure overall *system* performance, as the network performance can impact the packets injected into the network in systems. For example, the latency of the remote cache line can determine how long a processor is stalled and when the next packet can be injected into the network. In CMPs where processors, caches, and network are integrated onto a single chip, the NoC is much more closely coupled with processors and caches, compared to off-chip, large-scale networks. As a result, it is not clear if an open-loop evaluation is appropriate for on-chip networks.

Recent studies on interconnection networks have used alternative synthetic traffic models in their evaluation in order to complement the traditional open-loop evaluation [18], [4], [10], [16], [9]. Although closed-loop synthetic workloads have been previously used in these studies, the overall characteristics of closed-loop simulations have not been properly studied or evaluated. In this paper, we first evaluate the closed-loop synthetic workload model as a tool to evaluate the impact of NoC on the overall system. We then validate the accuracy of the closed-loop model against a detailed execution-driven simulation and extend the model to provide better correlation with the execution-driven simulation.

The contributions of this work are as follows:

- We compare open-loop and closed-loop measurements in on-chip networks and show how they mostly provide very similar results. Some differences between the two occur since the open-loop conventionally measures *average* performance while the closed-loop measures *worst-case* performance.
- We show how the closed-loop *batch model* can be extended with very simple models to mimic the behavior of the processor cores and the memory hierarchy. We demonstrate how these modifications increase correlation with results from execution-driven simulations.
- In an attempt to better correlate the batch model to the execution-driven simulation, we identify the impact of kernel traffic through full-system simulations on the overall system performance. We show how our simulation framework can be further extended to provide better correlation with full-system simulation results.

II. Background and Related Work

Different simulation methodologies have been used in the evaluation of interconnection networks. They can be classified into three types [8]: execution-driven simulation, trace-driven simulation, and synthetic workload simulation.

Execution-driven simulations involve modeling all aspects of a system, including the processing cores, memory hierarchy, and the interconnection network, and execute the application. An example of this methodology is Simics/GEMS[12][13], a full-system simulator with a detailed on-chip network model (GARNET [2]). Although execution-driven simulation provides accurate simulation results for a given application, the simulation is very time-consuming, even for a CMP with a small number of cores. As a result, it is not feasible for evaluating a large design-space of alternative architectures in a large-scale on-chip network.

An alternative to this approach is a *trace-driven simulation*, which replays a sequence of messages captured from

SC10 November 2010, New Orleans, Louisiana, USA 978-1- 4244-7558-2/10/$26.00

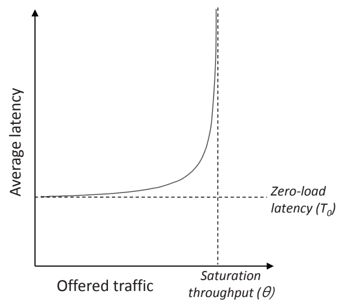


Fig. 1. Latency vs. offered traffic curve.

an execution-driven simulation on a network-only simulator. Since only abstract information of network packets such as the timestamp, packet size, and source and destination of packets is stored and replayed, the simulation can be performed significantly faster than an execution-driven simulation. However, since the traces are captured in advance, feedback from the network does not affect the workload and ignores the causality of messages. On the other hand, traces from applications running in a large number of cores on a chip are difficult to obtain, store, and manage, and typically it is not easy to vary the trace characteristics.

Another alternative is to use a network-only simulation with *synthetic workloads*, an approach that has been commonly used in the evaluation of interconnection networks. This method provides insights into the behavior of the network across a wide range of traffic patterns and provides a very fast simulation, thus enabling the design space exploration of the network. In this paper, we focus on synthetic workloads as a methodology for evaluating a large space of alternative on-chip networks in on-chip networks. We discuss how it can be extended to provide insights that are more reflective of not only *network* performance but also *system* performance.

A network simulation can be classified as either *open-loop* or *closed-loop* simulation.

A. Open-loop Measurement

Open-loop measurement [8] evaluates the network with traffic parameters, including spatial distribution, temporal distribution, and message size of packets, which are *independent* of the network itself – i.e., the network performance does not influence the traffic parameters. By using an infinite source queue, the traffic parameters are not influenced by the network. Using open-loop simulations, the network performance is often characterized by a latency/throughput curve (Figure 1) – for a specific traffic pattern, steady-state measurements are taken to plot the *average* packet latency. The zero-load latency (T_o), the maximum throughput (θ), and the impact of contention as the load increases are shown with this plot and provide measures of *network* performance.

B. Closed-loop Measurement

Closed-loop measurements differ from open-loop measurements in two ways: first, the feedback of the network impacts the simulation, and second, it attempts to measure *system* performance and not just *network* performance. For example, if the network buffers fill up, additional packets are not injected into the network. As a result, the network feedback impacts the overall performance measured. An execution-driven simulation is an example of a closed-loop simulation. In this work, we evaluate a synthetic workload with closed-loop measurement to evaluate the impact of the network on the overall system.

In a closed-loop synthetic workload, a pre-determined amount of "work" (which we refer to as the *batch size*) needs to be completed prior to termination of the simulation. The batch size attempts to model the amount of instructions [1] each core needs to execute a given application. The performance using closed-loop measurement is determined by the runtime to finish the work, defined by the batch size. Unlike an open-loop synthetic workload, the temporal aspect of the traffic pattern is influenced by the network, as the dependencies of messages impact the traffic injection rate. Based on the dependencies captured in the simulation, we classify synthetic closed-loop traffic simulations into two different types: closed-loop with *intra-node* dependency and closed-loop with *inter-node* dependency.

1) Closed-loop Model With Intra-node Dependency: Processors (and corresponding memory system) in a multicore processor have limited structural resources, thus restricting the amount of network traffic generated – for example, MSHRs (miss status hold registers [17], [11]) track the number of outstanding requests the processor can have. If the limit of MSHR is reached, further injection of new requests into the network is stalled until a reply is received from the network and the corresponding MSHR entry is removed. This model creates *intra-node* dependent traffic, as the injection of the current packet at a node is impacted by the network behavior of previously injected packets. This method has been used in prior work [10], [4], [15], [18] in the evaluation of on-chip networks, but the accuracy of this model has not been evaluated.

In this model, each node or processor has a fixed number of remote memory operations (e.g., remote cache line read/write requests), which we refer to as the batch size (b). For each request packet, once it reaches the destination, a reply packet is generated; we measure the runtime (T) required for all nodes to complete their operations. We also assume that each node has a limited number of outstanding memory operations or request packets (m) to model the effect of a MSHR – thus, if a node has injected m packets without having received any replies, it stalls until a response is received. The maximum number of packets in-flight per node (p_f) is m. We initially assume that a packet is injected in each node if $p_f < m$ and the batch size (b) has not been reached.

Figure 2 shows the impact of m and b in the batch model in a plot of b vs. runtime normalized to the batch size using the

[1]More specifically, the batch size represents load instructions that miss in the local cache; and thus necessitate accessing a remote cache or memory through the network.

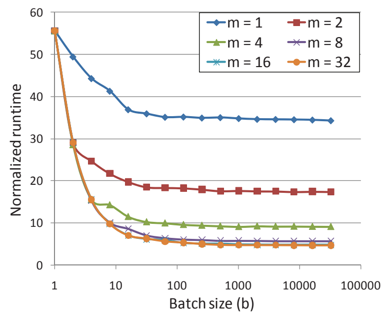
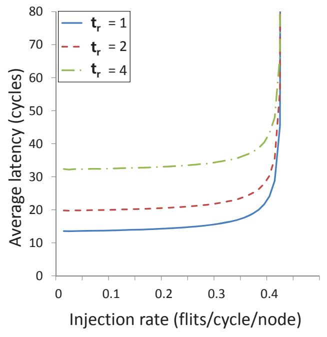
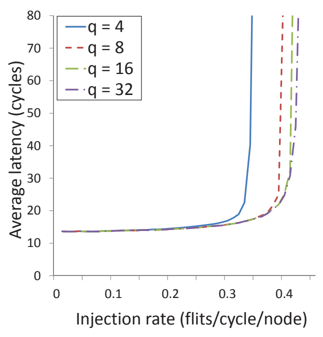


Fig. 2. Runtime normalized to the batch size in batch model.



(a) Router delay



(b) Buffer size

Fig. 3. Impact of router delay and buffer size in open-loop.

default values shown in Table I and the simulation described in Section III. As we increase b, the normalized runtime continuously decreases until the value saturates. A higher value of m results in lower normalized runtime, since it allows more requests to be overlapped, thus reducing the runtime *per* request or operation. The inverse of the asymptotic value that the runtime *per* request or operation approaches as the m value increases is the maximum throughput that can be achieved in the network. For the remainder of this paper, we use $b = 1000$ unless otherwise noted to obtain steady-state measurements.

2) Closed-loop Model With Inter-node Dependency: A barrier is a type of synchronization primitive where all threads/processes must wait until all the other threads/processes reach the barrier. These synchronization primitives create *inter-node* dependency, as the workload measurement is dependent on the behavior of other nodes in the network. Synthetic workloads using this model have been previously used, and include the barrier model [16] and burst-synchronized model [9].

Similar to the batch model, this measurement also assumes that all nodes have a fixed number of computations represented by a fixed number of communications (b). However, instead of intra-node dependency created with the m parameter, each node continues to inject packets into the network until (b) packets have been transmitted. Once every node has finished injecting packets and all injected packets have reached their destinations, the measurement is completed.

In this work, we focus on the batch model, instead of the barrier model, for two reasons. The barrier model essentially measures the throughput of the network and is very similar to open-loop measurements. In addition, it is anticipated that in future manycore processors, each core will not support a large number of outstanding requests without stalling the processor. In off-chip networks, for systems such as the Cray BlackWidow system [1] which can sustain over a thousand outstanding requests, this model might be representative, but for on-chip networks, it is expected that the processor will be able to tolerate only a handful of outstanding requests. In the rest of this paper, we focus on the batch model and discuss the similarities of the batch model to open-loop measurements. Then, in Section IV, we discuss how the batch model can be enhanced.

III. Evaluation of Closed-loop Model

In this section, we evaluate closed-loop measurements in an on-chip network described earlier in Section II-B1 and compare the results with those of open-loop measurements. The two methods use different performance metrics (average latency in the case of open-loop measurement, runtime for closed-loop measurements) but we show how the two different methods provide very similar insights into the network behavior. We compare the *relative* performance to evaluate the impact of different network parameters including router parameters, topologies, and routing algorithms. For some evaluations, we highlight how the results differ, because an open-loop evaluation focuses on *average* performance while the closed-loop measures *worst-case* performance. However, by using *worst-case* results from the open-loop, we show how similar trends can be found in the two evaluations.

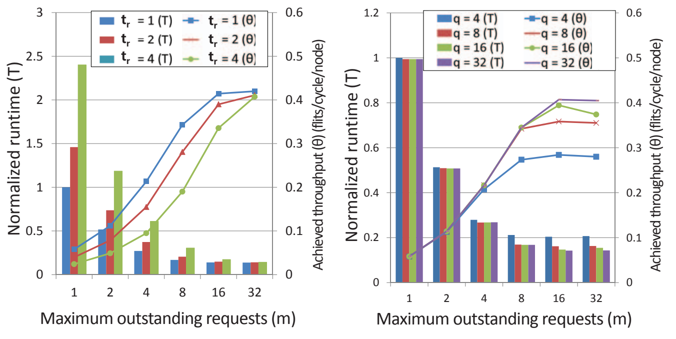
A. Simulation environment

We use a cycle-accurate network simulator [8] to evaluate a 64-node on-chip network using an 8-ary 2-cube (2D mesh) topology. A 256-node on-chip network using a 16-ary 2-cube topology is also evaluated, but the results are not included as they show a similar trend. The parameters used in the evaluation are described in Table I and bold values show our baseline.

TABLE I
Simulation Parameters

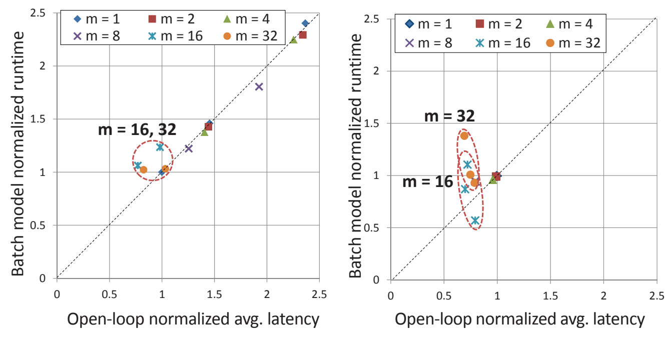
Topology	8x8 2D mesh, 16x16 2D mesh
Virtual channels VC buffer size	2, **4** 1, 2, 4, 8, **16**
Router delay (cycle) Routing algorithm Arbitration	**1**, 2, 4, 8 **Dimension ordered routing(DOR)**, Valiant (VAL), Min. adaptive(MA), ROMM [14] **Round robin**, age-based
Link delay Link bandwidth	1 cycle 1 flit/cycle
Packet sizes Traffic patterns	**1 flit**, bimodal (1 flit and 4 flit) **uniform random**, bit reversal bit complement, transpose

The following symbols are used in our description of network parameters and simulation parameters in the following



(a) Router delay (b) Buffer size

Fig. 4. Impact of router delay and buffer size in batch model.



(a) Router delay (b) Buffer size

Fig. 5. Comparison of impact of router delay and buffer size between open-loop simulation and batch model.

sections.

t_r	: router delay
q	: queue size
N	: network size
b	: number of requests/operations that needs to be executed per node (batch size)
m	: number of maximum outstanding requests/operations per node
p_f	: number of request packets in flight per node
T	: total runtime
p	: total number of packets ($p = N \times b \times 2$)

B. Experiment 1: Router parameters

We first compare the two simulation method as we vary two router parameters, router delay (t_r) and buffer size (q). As we increase t_r, the zero-load latency of the open-loop simulation increases while similar saturation throughput is achieved (Figure 3(a)). For example, by increasing t_r from 1 to 2 and 4, the zero-load latency increases by ratios of 1.5 and 2.5, respectively. The expected increase in zero-load latency is not 2 or 4, since for each hop, the channel delay is added, thus resulting in zero-load latency increases of only 1.5 and 2.5. However, regardless of t_r, the saturation throughput does not change, as the network saturates at approximately 43%. The performance impact of buffer depth (q) is shown in Figure 3(b) for two virtual channels with single-flit packets. Simulations using different packet sizes (such as a mixture of short and long packets) did not impact the comparisons. The buffer depth (q) does not impact zero-load latency but limited buffer depth impacts the throughput. For example, with $q = 4$, the throughput can be reduced by approximately 15.5%, compared to a network with $q = 16$. Continuing to increase the buffers beyond 16 results in limited improvement in throughput, as the buffers no longer become the network bottleneck.

The results in Figure 3 are well understood, and based on these open-loop measurements, we obtain insights into the impact of t_r and q on network performance as load increases. To evaluate the differences between open-loop and closed-loop measurements, we compare the results of Figure 3 to a closed-loop simulation model as we vary the same two parameters (t_r, q), which are shown in Figure 4. We plot the runtime normalized to the runtime with $t_r = 1$, $q = 32$, and $m = 1$. The achieved throughput (θ) of the workload is calculated from the runtime ($\theta = (b \times 2)/T$).

Interestingly, simulation results from the closed-loop model show very similar trends with the open-loop simulation. For small values of m, the achieved throughput is relatively low, and increasing t_r also increases T with a ratio that is similar to the increase in zero-load latency (Figure 4(a)). For large values of m, where the achieved throughput approaches the saturation throughput, the impact of t_r is nearly negligible – similar to the different values of t_r resulting in the same throughput using an open-loop simulation. A similar trend can be observed for the impact of buffer size – at near zero-load (or small values of m), there is minimal impact on the overall performance but as the load increases with higher values of m, larger buffer size improves the network performance.

Figure 5 shows the scatter plot comparing the normalized runtime of the batch model and normalized average latency obtained from the open-loop measurements. The scatter plot data points are obtained from the following steps.

1) Run the batch model from Section II-B1 with $b = 1000$ and different values of m and network parameters (t_r,q).
2) Calculate the *achieved* throughput of the batch simulations using $\theta = (b \times 2)/T$.
3) Run the open-loop simulation with an offered load equal to θ for each of the values of m and (t_r, q).
4) Plot the normalized batch runtime vs. the normalized latency measured for the open-loop simulation. [2]

The scatter plot shows a high correlation between the batch model and open-loop simulations as a correlation coefficient of 1 indicates a highly correlated relationship. However, for several data points from a large m, poor correlation can be found. Since we are measuring the average latency of open-loop simulations near the saturation throughput, the actual latency is highly sensitive to the offered load and its high

[2]To show the correlation across the different values of m on a single plot, we normalize the values of both the batch model and the open-loop simulations to the baseline result of *each* m – e.g., for simulations where we vary t_r, all results of $m = 1$ are normalized ($m = 1$, $t_r = 1$), all results of $m = 4$ are normalized ($m = 4$, $t_r = 1$), etc. As a result, normalized latency of $m = 16$ can be lower than $m = 1$.

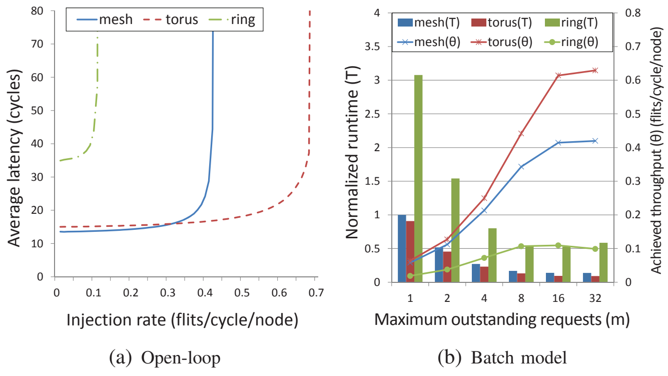


Fig. 6. Impact of topology in open-loop and batch model simulation.

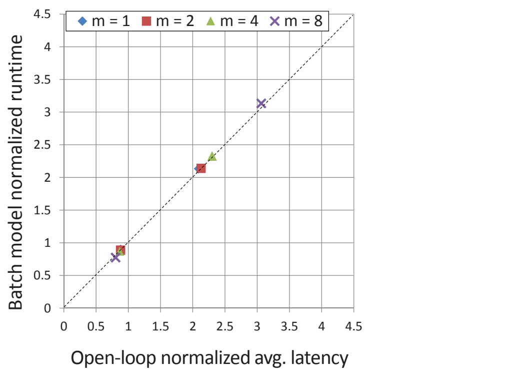


Fig. 8. Correlation between open-loop and batch model under different topologies.

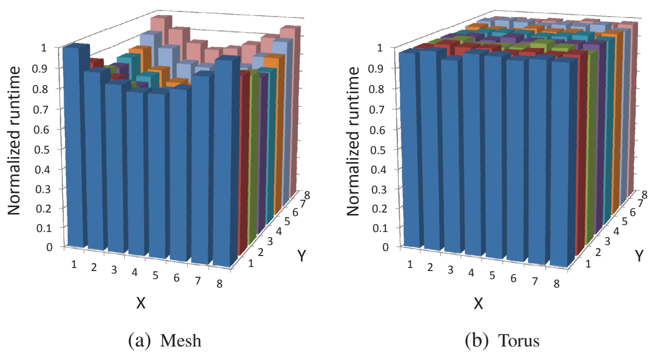


Fig. 7. Runtime of each node under mesh and torus.

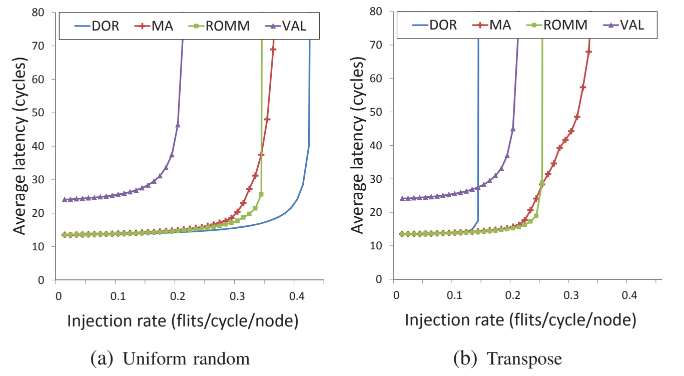


Fig. 9. Impact of routing algorithm in open-loop simulation.

variation results in inaccuracy of comparison. [3] As a result, if we exclude the results for $m = 16$ and 32, the correlation coefficient is calculated to be 0.9953 for the evaluation of t_r and 0.993546 for the evaluation of q. Thus, with network parameters such as t_r and q, both open-loop and closed-loop measurements have very similar impact on performance, even though the two measurements use different performance metrics.

C. Experiment 2: Topology

In this section, we compare the two simulation models with different topologies. We evaluate a mesh, ring, and torus on 64 nodes using the parameters in Table I and the results are shown in Figure 6 for a uniform random traffic pattern. [4] Open-loop simulation results show the ring topology has the highest latency and lowest throughput for 64 nodes. The torus shows a slightly higher zero load latency than the mesh since we assume a folded-torus [8], which increases the channel delay. The torus achieves higher throughput since it has the highest bisection bandwidth.

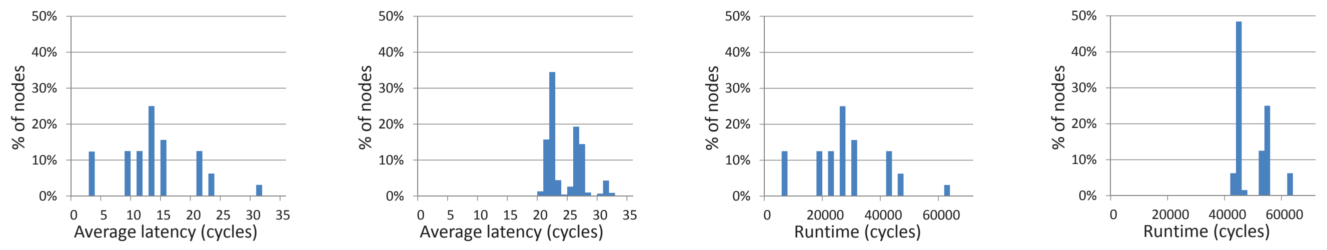
Similar trends are also shown in Figure 6(b) with the closed-loop model. However, for a small m, the mesh shows higher runtime than the torus, a finding that is different from the open-loop measurements. Because the mesh is not an edge-symmetric topology, the closed-loop model shows that the nodes near the center of the network finish much faster than the outer nodes (Figure 7(a)). In comparison, the edge-symmetric torus topology shows very similar runtime for all nodes (Figure 7(b)). As a result, the mesh results in higher runtime even with lower average packet latency. Figure 8 shows a scatter plot of the normalized runtime of the batch model versus the normalized *worst-case* latency of the open-loop. The use of worst-case latency (instead of average latency) in the open-loop simulation results in a very similar impact of the network parameter on the overall performance, and the correlation coefficient is calculated to be 0.999.

D. Experiment 3: Routing algorithms

We evaluate the impact of routing algorithms with different traffic patterns including uniform random and transpose patterns. Other traffic patterns including bit reversal and bit complement were simulated but follow a similar trend and are not included due to space constraint. The 2D mesh network described in Table I was used and the performance results are shown in Figure 9 for an open-loop and Figure 10 for a closed-loop. Similar to the impact of router parameters in Section III-B, the two results show almost the same trend – at low m, the results of the closed-loop follow the zero-load latency trend of the open-loop while at high m, the saturation

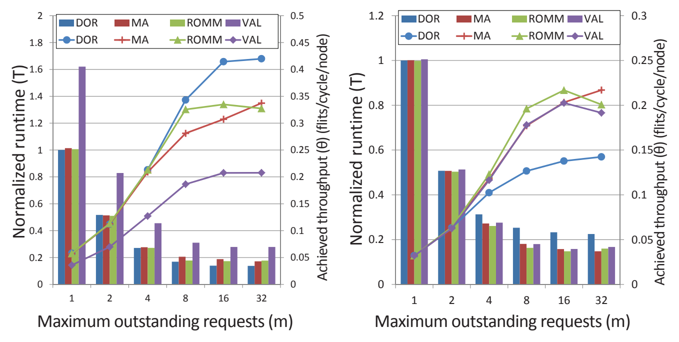
[3]The saturation throughput is defined as the load where latency approaches infinity [8] – thus it is difficult to obtain an accurate latency value.

[4]For a more accurate topology comparison, the bisection bandwidth should be held constant; however, for simplicity, the *channel* bandwidth is held constant in our comparison.



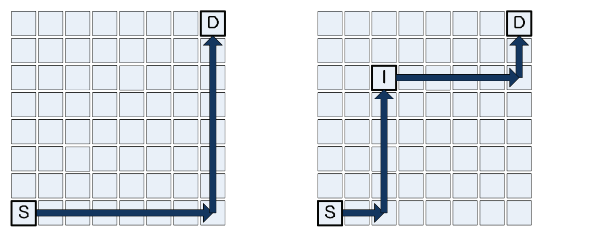
(a) DOR (b) VAL (c) DOR (d) VAL

Fig. 11. Node distribution of (a)(b)average latency in open-loop simulation and (c)(d)runtime in batch model.



(a) Uniform random (b) Transpose

Fig. 10. Impact of routing algorithm in batch model.



(a) DOR (b) VAL

Fig. 12. Example routing using (a) DOR and (b) VAL. S and D represent the source and destination node and I represents the intermediate node in VAL.

throughput trends are seen in the closed-loop.

However, there is a noticeable difference between the open-loop and closed-loop with Valiant routing (VAL) under some permutation traffic patterns such as transpose traffic. The latency-throughput curve in Figure 9(b) for the transpose patterns shows VAL resulting in higher zero-load latency but also higher throughput, since VAL exploits path diversity to load-balancing compared to DOR. However, for $m = 1$ in the batch model (Figure 10(b)), there is negligible impact of higher zero-load latency, as VAL results in only a 1.7% increase in runtime compared to DOR. This discrepancy highlights the difference between the two evaluation models. Open-loop simulations measure *average* network performance through average packet latency [5] while the closed-loop measurement is dependent on *worst-case* performance, as the runtime is decided by the node with the largest runtime.

To better understand the difference, the average latency distribution of the open-loop simulation with DOR and VAL are shown in Figure 11(a,b) and the runtime distributions of the batch model with DOR and VAL are shown in Figure 11(c,d) for $m = 1$. The *average* runtime for DOR is lower than VAL by 44% – similar to the zero-load latency difference in Figure 9(b). However, the worst-case runtime is identical, as they are determined by the nodes located in the corner for the transpose traffic pattern (Figure 12). For traffic between these two nodes in a 2D mesh network, VAL routing still results in *minimal* routing – resulting in an identical worst-case zero-load between DOR and VAL. As a result, the routing algorithm has minimal impact on such traffic patterns if the worst-case measurement is used for a low injection rate.

[5]The saturation throughput does measure the worst-case performance, since the throughput is determined when one channel in the network is saturated.

IV. Batch Model Validation

In this section, we attempt to validate the accuracy of the batch model against an execution-driven simulation and highlight the shortcomings of the proposed model in measuring overall *system* performance. Through very simple extensions, we show how the batch model can be improved to better correlate with execution-driven simulations.

A. Evaluation Methodology

We compare our batch model against an execution-driven simulation using a Simics [12]/GEMS [13] simulator, using the parameters shown in Table II. We evaluate a 16-core CMP system with each core consisting of an in-order SPARC processor. Garnet [2] network is used, as the 16 nodes are interconnected with a 4x4 2D mesh topology and a 1-cycle router delay is used as a baseline simulation. Similar to the simulations in Section III-B, we vary the router delay to evaluate its impact on overall performance. We used the `barnes`, `fft`, and `lu` benchmarks from SPLASH-2 [19] and `blackscholes` and `canneal` applications from the PARSEC [6] benchmark suite. The benchmarks were warmed up and checkpointed to avoid cold-start effects. We compared the *relative* performance or *speedup* as the network parameters were changed and compared these changes to our batch model. The GEMS simulation is obviously a more detailed simulator than a network-only simulator. However, because of this additional complexity, the simulation can take on the order of days to run SPLASH2 or PARSEC benchmarks.

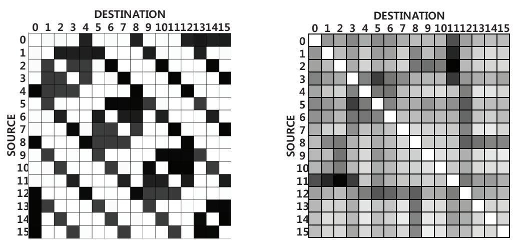


Fig. 13. (a) Communication pattern of `lu` and (b) actual traffic communication pattern.

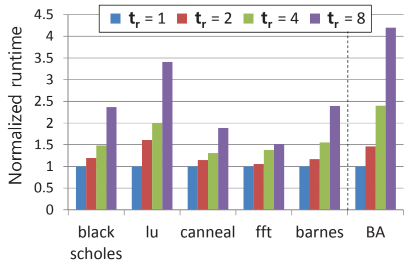


Fig. 14. Normalized runtime of GEMS+Garnet and batch model (BA) as router delay (t_r) is varied.

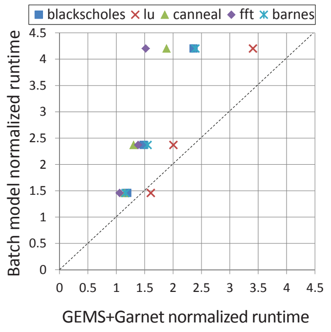


Fig. 15. Correlation between GEMS+Garnet and batch model

For example, simulation of `barnes` using medium-size input using GEMS took 88.5 hours on a Intel Xeon 2.53 GHz core. A prior study [20] has shown that adding out-of-order modeling capabilities to GEMS (through opal module) slows down the simulations by a factor of almost 17. But with a network-only simulation using a synthetic workload, the simulation time can be significantly reduced, as it takes only a few minutes to simulate a 64-node network.

TABLE II
SIMICS/GEMS+GARNET SIMULATION PARAMETERS

Processor	16 in-order SPARC cores
L1 Caches	Split I&D, 32 KB 4-way set associative, 2 cycle access time, 64-byte line
L2 Caches	shared L2, 512KB per tile (total 8MB) 10 cycle access time, 64-byte line
Memory	300-cycle DRAM access time
On-chip Network	4-ary 2-cube mesh, 16-bytes links, **1**/2/4/8 router delay, 8 VCs, 4 buffers/VC 1 cycle on-chip link latency, DOR

For comparison with GEMS+Garnet, we use uniform random traffic for the batch model. Prior work [5] has shown explicit communication patterns for different benchmarks; if we attempt to characterize the explicit communication between CPUs in the application, we observe a communication pattern as shown in Figure 13(a) for the `lu` benchmark. This figure plots the spatial communication between the nodes and darker squares correspond to heavier traffic. However, if we observe the actual traffic injected from the nodes rather than just the communication pattern inherent in the application, the communication distribution shown in Figure 13(b) is observed where the traffic appears more random. Thus, we use uniform random traffic in comparing the batch model to the execution-driven simulation.

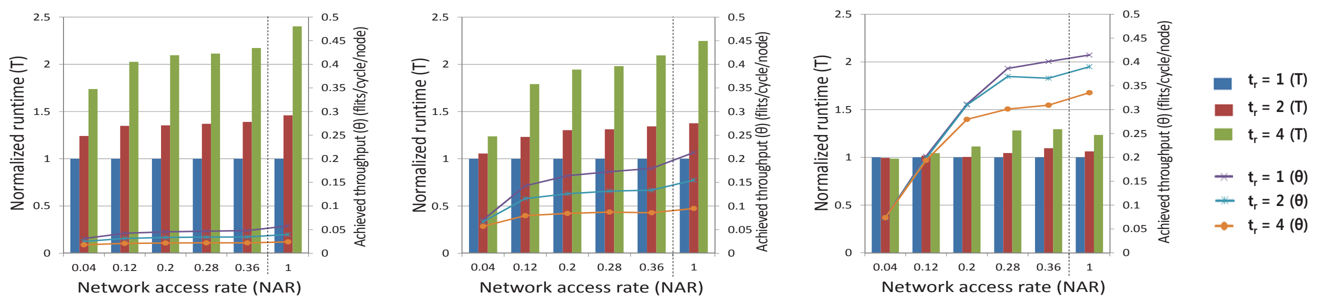
B. Evaluation Results

Figure 14 illustrates the impact of the router delay on the runtime of the batch model (BA) and the real workloads on GEMS+Garnet, which are normalized to our baseline setup. In this figure, each benchmark is impacted differently by an increase in router delay. For example, while the runtime of `lu` increases by more than 3× as the router delay increases from 1 to 8, the runtime of `fft` is only increased by 1.51×. However, the batch model does not reflect any difference between these benchmarks. As shown earlier in Section III-B, by increasing the router delay from 1 to 2, 4, and 8, the runtime from the batch model increases by ratios of approximately 1.45, 2.4, and 4.2. As a result, a poor correlation between the performances of the batch model and the execution-driven simulation is shown in Figure 15, as the correlation coefficient was calculated to be 0.829.

Simply modeling just the MSHR with the batch model does not provide accurate modeling of the different benchmarks, as the batch model is limited in providing insights into network-only performance. To close the gap between the batch model and the execution-driven simulation, we propose several simple extensions to the batch model in an effort to provide better insights into the impact of the network on the overall system performance without significantly increasing the simulator complexity or decreasing the simulation speed.

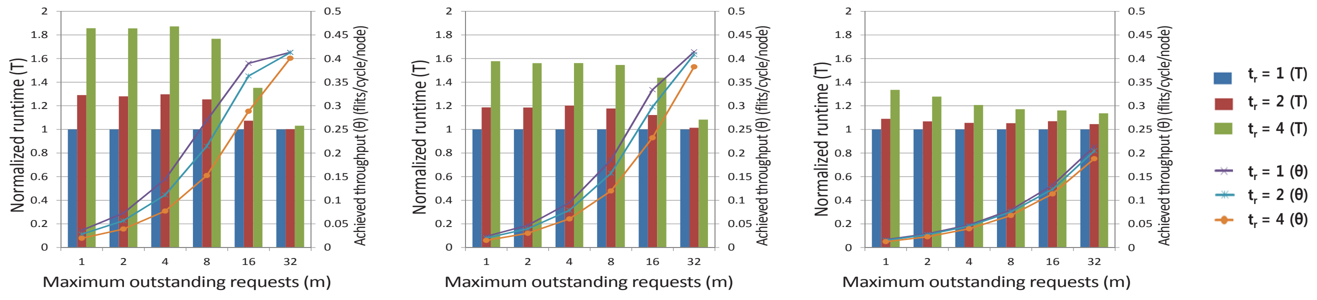
C. Enhancing the Batch Model

In this section, we propose very simple techniques to extend the batch model so as to provide better correlation between the network performance and overall system performance. We extend the batch model by modeling the processing core and the memory hierarchy to change the actual injection rate of packets into the network. Although packets are generated by complex interactions between the processor architecture, cache hierarchy, and memory system, the addition of simple models to adjust the injection rate can provide a better understanding of the impact of the network parameters on the overall system performance. The two simple modifications we propose are adding the injection model of request packets and adding



(a) m = 1 (b) m = 4 (c) m = 16

Fig. 16. Evaluation of the batch model with an enhanced injection model.



(a) Memory latency = 20 (b) Memory latency = 50 (c) Memory latency = 20 + 0.1 * 300

Fig. 17. Evaluation of the batch model with an enhanced reply model.

latency before reply packets are injected into the network. [6]

1) Enhanced Injection Model: In the baseline batch model, additional packets are always injected into the network if $p_f < m$. This injection policy might be representative of applications that are communication bound with a high-network demand but is not representative of other applications. For example, as the L1 miss rate is often under 10%, accessing a shared L2 might be infrequent and would result in a lower network injection rate [7]. In addition, data dependencies can stall the execution of the processor, which will also affect the packet injection rate. To model this effect, we define the parameter network access rate (NAR) as the packet injection rate of an application under an *ideal* on-chip network. [7] The NAR attempts to model the network usage of different applications by characterizing the injection rate when the network does not impact the workload. With this addition to the batch model, if $p_f < m$, additional packets are only injected at a rate of NAR, and if $p_f = m$, no additional requests are injected.

The synthetic workloads used in Section III-B are re-evaluated with the addition of the NAR injection rate and are shown in Figure 16 for different values of NAR as t_r is varied. NAR $= 1$ corresponds to the baseline batch model described earlier in Section III. As the value of NAR decreases, the impact of the network parameter on the overall performance *decreases* while as NAR increases, the performance comparison approaches that of the baseline batch model.

Interestingly, for large values of m ($m = 16$) and small values of NAR, the overall injection rate is low, and based on results of Section III-B, the performance difference should correspond to the increase in zero-load latency. However, the router latency has minimal impact on the overall performance. Since these data points represent non-communication limited workloads (i.e., the maximum value of m is not reached with small NAR values), the router latency – although it increases *network* latency – has minimal impact on overall performance. From this simple extension to the baseline model, we can see how the impact of the network on the overall performance is reduced by modeling the injection rate through NAR.

2) Enhanced Reply Model: In our baseline batch model, when requests arrive at their destination, corresponding reply packets are generated immediately and injected into the network. However, in actual CMP systems, the reply packets will not be immediately injected but will have an added delay resulting from either a L2 cache access or main memory access. In this section, we evaluate the impact of this additional latency in the batch model using two simple models: a fixed latency model that adds a fixed latency for all remote memory accesses and a probabilistic model that adds L2 access latency for a given L2 hit rate or adds L2 access latency + main memory access latency otherwise. Figure 17 demonstrates the result of these models. We use 20 cycles as L2 access latency, and 300 cycles as memory latency; in Figure 17(c) it is assumed that 10% of the requests need to access the memory.

[6]These extensions cannot be applied to open-loop simulations, since injection of the packets cannot be affected by the network or other components.

[7]An ideal network is defined as a fully connected network with infinite bandwidth between the nodes and single cycle latency.

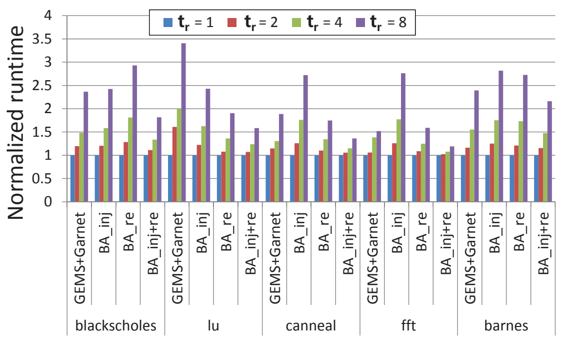


Fig. 18. Normalized runtime of GEMS+Garnet and batch model.

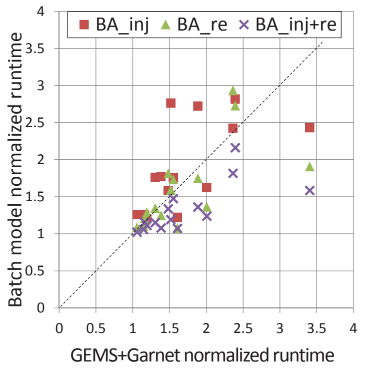


Fig. 19. Correlation between GEMS+Garnet and batch model.

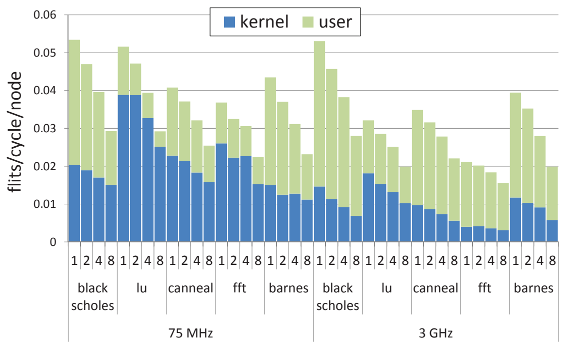


Fig. 20. Network injection rate of benchmarks in GEMS+Garnet as the router delay is varied.

With this simple addition to the model, we model the obvious effect that as the memory access latency increases, the impact of the router delay is reduced, as the overall remote access latency is dominated by the memory latency itself.

However, interesting observations can be made in Figure 17(b,c). Both results use the same *average* memory latency $20 + 0.1 * 300 = 50$ but Figure 17(c) shows a lower injection rate and reduced impact of the router latency. Although the average memory latency is the same, Figure 17(c) includes the effect of having long memory operations and reduces the impact of the router latency on the overall performance.

D. Comparison with GEMS+Garnet

Based on the two simple extensions described in the previous sections, we again compare the results of execution-driven simulations with the extended batch model, that is, BA_injection(BA_inj), BA_reply(BA_re), and BA_enhanced(BA_inj+re), which include the additional modeling of both the injection and the reply. We evaluate the correlation of the modified batch model as the router delay is varied (Figure 19). The performance results are normalized to the baseline result of $t_r = 1$. The NAR values are calculated and are shown in Table III. Although better correlation can be seen than that obtained using the baseline batch model, there is still a discrepancy with the results of the execution-driven simulations. Notably, BA_inj+re, a more enhanced model, shows worse correlation than BA_inj or BA_re, contrary to our expectations. In the next section, we identify how the *kernel* traffic from the OS impacts the network traffic and the overall performance, which provides the reason for the poor correlation of BA_inj+re, and also describe how the batch model can be extended.

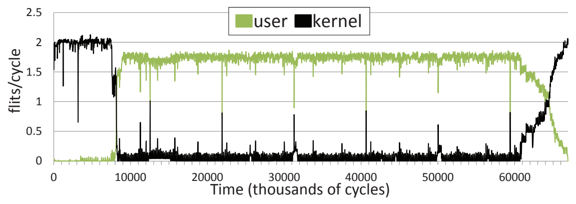
TABLE III
GEMS SIMULATION CALCULATION OF NAR

Benchmarks	Ideal cycle count	Total flits	NAR	L2 miss rate
`blackscholes`	44,228,000	39,576,862	0.028	0.006
`lu`	247,498,080	86,601,157	0.011	0.183
`canneal`	70,915,759	90,944,651	0.040	0.207
`fft`	139,433,783	147,472,376	0.033	0.629
`barnes`	501,330,834	753,434,335	0.047	0.019

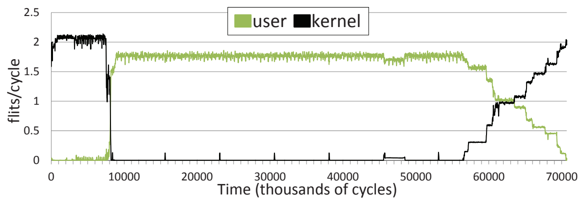
V. Modeling Kernel Effect

As identified in prior work [3], the operating system (OS) can have significant impact on the variability of simulation results. To improve the accuracy of our batch model compared to execution-driven simulation results, we investigate the effect of the OS on the network traffic and the overall performance. The kernel activities that are found in the applications we evaluated can be classified into two types: traps or system calls invoked by the applications and periodic timer interrupts. These two types of events have very different impact on the network traffic – the network traffic from the timer interrupt handler is proportional to the simulation runtime while the traffic from traps or system call handlers is independent of the runtime.

Figure 20 shows how a significant portion of the network traffic is generated by the kernel activities. We show the results for two different core clock frequencies, 75MHz and 3GHz, where 75MHz is the default clock frequency in the Simics configuration for a Serengeti server [12] while 3GHz represents a clock frequency of a modern high-end processor. For benchmarks such as `lu`, the kernel traffic account for more than 80% of the total network traffic. In addition, the ratio of the kernel traffic is much higher with the 75MHz clock than the 3GHz clock. The effect of periodic timer interrupts is much more significant with 75MHz than with 3GHz configuration, as the interval of the timer interrupts is influenced by the wall clock time, not the number of simulation cycles. For

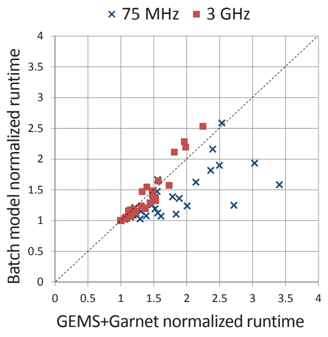


(a) 75 MHz

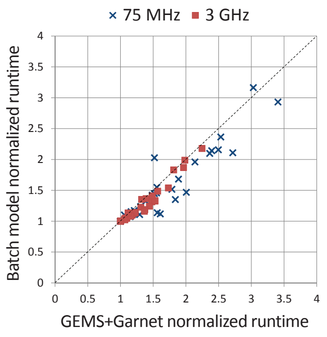


(b) 3 GHz

Fig. 21. Injection rate of `blackscholes` in GEMS+Garnet.



(a) Without OS model



(b) With OS model

Fig. 22. Correlation between GEMS+Garnet and batch model with/without OS modeling.

example, in `blackscholes`, as we vary t_r, the numbers of timer interrupts that we counted are 6, 6, 8, and 10 in a 3GHz simulation but increase to 422, 495, 575, and 915 in a 75MHz simulation.

In Figure 21, we plot the overall injection rate of the 16 cores for `blackscholes` benchmark with 75MHz and 3GHz clock frequency in Simics to identify the two types of kernel activities more clearly. For the periodic timer interrupts, we can see six small peaks in the 3GHz simulation while there are significantly larger numbers of small peaks in the 75MHz simulation. In addition, we can also identify a significant amount of kernel traffic at the beginning and the end of the simulation in Figures 21 – which is caused by system calls such as thread creation or synchronization. This additional network traffic from the kernel activities needs to be incorporated into the batch model by adjusting the batch size.

To incorporate the network traffic from the kernel activities, we increase the batch size in two ways. The batch size is statically increased (prior to simulation) to represent the application dependent additional traffic from system calls or traps. In addition, the batch size is dynamically increased to represent the additional traffic from periodic timer interrupts. After determining the rate of the periodic timer interrupt (R_{timer}) from the execution-driven simulations, the actual additional traffic generated in the batch model is dependent on the batch model simulation runtime – i.e., additional packets or "batch" is injected into the network approximately every $1/R_{timer}$ cycles and the total amount of packets injected in the simulation is proportional to the batch model simulation runtime.

TABLE IV
CHARACTERISTICS OF BENCHMARKS

Benchmarks	NAR		L2 miss rate		Application dependent additional traffic	R_{timer}
	user	OS	user	OS		
`black scholes`	0.024	0.266	0.004	0.013	0.58	0.00245
`lu`	0.021	0.048	0.418	0.005	0.53	0.0080
`canneal`	0.038	0.126	0.274	0.029	0.57	0.0038
`fft`	0.033	0.442	0.708	0.021	0.34	0.0056
`barnes`	0.055	0.063	0.011	0.017	0.67	0.0015

With these additional modeling, Figure 22 shows the correlation plot of the normalized runtime of the enhanced batch model versus the normalized runtime measured using GEMS+Garnet. By incorporating this additional modeling of the kernel traffic, the correlation coefficient is improved to 0.9724 (compared to 0.954136 without any additional modeling of the kernel traffic) for the 3GHz simulation; meanwhile, for the 75MHz simulation, significant correlation improvement can be seen, from 0.705246 in Section IV to 0.9311.

VI. Conclusion

On-chip networks are different from off-chip networks, as the processors, memory, and the network are closely coupled together. In this work, we show how conventional open-loop simulations can provide very similar results as closed-loop, batch model simulations. However, we also show how open-loop simulations can provide misleading results as it only measures *network* performance and not *system* performance. We extend the batch model with a simple methodology that incorporates the impact of the processor and the memory system on the simulation results. As a result, different insights into the impact of the network parameters compared to open-loop measurements are obtained. The proposed methodology is intended to help architects with the evaluation of on-chip network design and facilitate design space exploration of future on-chip network designs.

Acknowledgement

The authors would like to thank the anonymous reviewers, Brian Towles, and James Balfour for their comments. This work was supported in part by the IT R&D Program of MKE/KEIT [2010-KI002090, Development of Technology Base for Trustworthy Computing] and in part by Microsoft Research Asia (MSRA) New Faculty Fellowship.

References

[1]D. Abts, A. Bataineh, S. Scott, G. Faanes, J. Schwarzmeier, E. Lundberg, T. Johnson, M. Bye, and G. Schwoerer. The Cray BlackWidow: a highly scalable vector multiprocessor. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007.

[2]N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha. GARNET: A detailed on-chip network model inside a full-system simulator. In *ISPASS*, pages 33–42, 2009.

[3]A. R. Alameldeen and D. A. Wood. IPC considered harmful for multiprocessor workloads. *IEEE Micro*, 26(4):8–17, 2006.

[4]J. Balfour and W. J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *ICS '06: Proceedings of the 20th annual international conference on Supercomputing*, pages 187–198, New York, NY, USA, 2006.

[5]N. Barrow-Williams, C. Fensch, and S. Moore. A communication characterisation of Splash-2 and Parsec. In *IISWC '09: Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, pages 86–97, Washington, DC, USA, 2009. IEEE Computer Society.

[6]C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC benchmark suite: characterization and architectural implications. In *PACT '08: Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81, New York, NY, USA, 2008. ACM.

[7]S. Cho and L. Jin. Managing distributed, shared l2 caches through os-level page allocation. In *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 455–468, Washington, DC, USA, 2006. IEEE Computer Society.

[8]W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA, 2003.

[9]C. Izu, J. Miguel-Alonso, and J. A. Gregorio. Evaluation of interconnection network performance under heavy non-uniform loads. In M. Hobbs, A. M. Goscinski, and W. Zhou, editors, *ICA3PP*, volume 3719 of *Lecture Notes in Computer Science*, pages 396–405. Springer, 2005.

[10]J. Kim, W. J. Dally, and D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 126–137, New York, NY, USA, 2007.

[11]D. Kroft. Lockup-free instruction fetch/prefetch cache organization. In *ISCA '81: Proceedings of the 8th annual symposium on Computer Architecture*, pages 81–87, Los Alamitos, CA, USA, 1981. IEEE Computer Society Press.

[12]P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, 2002.

[13]M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *SIGARCH Comput. Archit. News*, 33(4):92–99, 2005.

[14]T. Nesson and S. L. Johnsson. ROMM routing on mesh and torus networks. In *SPAA '95: Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, pages 275–287, New York, NY, USA, 1995.

[15]Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary. Firefly: illuminating future network-on-chip with nanophotonics. In *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*, pages 429–440, New York, NY, USA, 2009.

[16]A. Singh. *Load-Balanced Routing in Interconnection Networks*. PhD thesis, Stanford University, 2005.

[17]J. Tuck, L. Ceze, and J. Torrellas. Scalable cache miss handling for high memory-level parallelism. In *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 409–422, Washington, DC, USA, 2006. IEEE Computer Society.

[18]S. W. Turner. Performance analysis of multiprocessor interconnection networks using a burst-traffic model. Technical report, Champaign, IL, USA, 1995.

[19]S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: characterization and methodological considerations. In *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*, pages 24–36, New York, NY, USA, 1995. ACM.

[20]H. Zeng, M. Yourst, K. Ghose, and D. Ponomarev. MPTLsim: a cycle-accurate, full-system simulator for x86-64 multicore architectures with coherent caches. *SIGARCH Comput. Archit. News*, 37(2):2–9, 2009.