# SigNet: Network-on-Chip Filtering for Coarse Vector Directories

Natalie Enright Jerger
Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON
enright@eecg.toronto.edu

***Abstract*—Scalable cache coherence is imperative as systems move into the many-core era with cores counts numbering in the hundreds. Directory protocols are often favored as more scalable in terms of bandwidth requirements than broadcast protocols; however, directories incur storage overheads that can become prohibitive with large systems. In this paper, we explore the impact that reducing directory overheads has on the network-on-chip and propose SigNet to mitigate these issues. SigNet utilizes signatures within the network fabric to filter out extraneous requests prior to reaching their destination. Overall, we demonstrate average reductions in interconnect activity of 21% and latency improvements of 20% over a coarse vector directory while utilizing as little as 25% of the area of a full-map directory.**

## I. Introduction

With transistor counts continuing to scale with Moore's Law, chip architects are able to integrate an increasing number of cores on chip. As parallel architectures remain largely reliant on a shared memory paradigm, cache coherence protocol implementations and the resultant communication will become a significant performance bottleneck. Efficient handling of coherence actions within the network-on-chip (NoC) is imperative to continue scaling many-core architectures.

Various coherence schemes have been explored over the years; the majority fall into two categories: broadcast and directory coherence. These coherence protocols can be characterized by their *bandwidth* requirements, their *latency* to satisfy requests and their *storage* overhead requirements.

Broadcast protocols exhibit low latency but are limited to a small number of nodes due to high bandwidth requirements that quickly saturate a bus. Broadcast protocols do not incur storage overheads. Alternatively, directory protocols have been favored for large-scale multiprocessors as they require lower bandwidth allowing greater scalability. A disadvantage of directory protocols is the large storage overheads required to maintain sharing lists for cached memory blocks.

Directory protocols pose significant storage challenges moving forward toward massively many-core architectures. Well-known techniques to address the storage overhead can be employed [3], [9], [11]; however, these techniques in turn increase the bandwidth demands that a directory protocol places on the NoC. Coarse vector (CV) directories have been employed to reduce storage overhead without resorting to full broadcast in the common case [11]. Work by Gupta et. al [11] utilizes the notation $Dir_iCV_r$ where $i$ is the number of pointers and $r$ is the number of cores in a region represented by a single bit in the coarse vector.

When the number of sharers of a cache block exceeds $i$ causing the pointers to overflow, the directory entry will operate in a coarse mode. CV directories use one bit in the sharing vector to represent a region of cores (multiple cores map to the same bit). When an invalidation is required, extraneous processors will receive and acknowledge this invalidation request due to the imprecise sharing list. Extraneous invalidation requests also result in additional cache accesses which wastes power and can delay non-extraneous requests. CV directories can also result in additional bandwidth to locate a block on-chip for cache-to-cache transfers.

For many applications commonly studied, the number of cores sharing a cache line is small (e.g. close to 2); more widely shared lines exist but occur infrequently in most applications [13]. Previous work in CV directories has exploited this property [11]. Assuming such sharing trends continue as applications scale to many-core architectures, a directory organization with two or four pointers could be a feasible design point that does not suffer from frequent pointer overflows. For example, with a 256-core architecture, 2 cores are represented with two 8-bit pointers; when the two pointers overflow, the coarse vector would have 16 cores per region. 16 invalidation requests would be generated.

Ideally, an NoC design cognizant of CV directories would attempt to eliminate extraneous invalidation requests (requests that go to invalid cache blocks). SigNet utilizes Bloom filters [4] inside each router to summarize cache contents. Addresses that map to valid bits in the Bloom filter *may* be cached at downstream routers. Messages pertaining to these addresses must continue their network traversal. Addresses that map to invalid bits in the Bloom filter *do not* reside in downstream caches and their network traversal can be safely terminated. SigNet reduces NoC activity by an average of 21%, average packet latency by 20% and invalidation completion time by 22% compared to a CV directory with no NoC support.

SigNet is a novel router microarchitecture that provides an NoC with *tightly-coupled* coherence filters to reduce NoC pressure. This paper makes the following contributions:

- Characterizes the impact of CV directory architectures on both the latency and bandwidth of the NoC.
- Proposes in-network filtering to reduce the load on the NoC to improve performance and save power.

## II. Motivation

In this section, we first compare the overheads of full-map directory vectors to CV directories to support our claim that CV directories will be necessary (from a storage perspective) for large core counts. Next, we discuss the negative impacts on NoC latency and bandwidth that result from utilizing coarse vectors. Finally, we present data illustrating the potential for negative system side effects due to coarse vectors.

This research was supported in part by an NSERC Discovery grant. The author would like to thank Andreas Moshovos, Jason Zebchuk and the anonymous reviewers for their suggestions on improving this work.

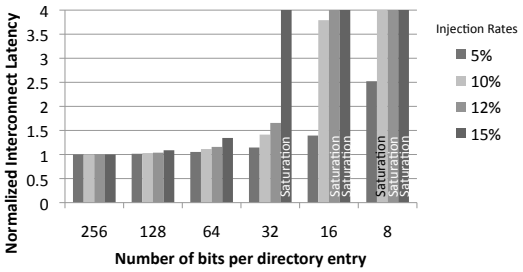



Fig. 1: Normalized average packet latency as the number of bits in the sharing vector decreases

**Storage Overheads**. A full-map directory and 256 cores on-chip requires 32 bytes of overhead per directory entry to identify the sharers. Per 64-byte cache line, this represents an unacceptable 50% overhead for just the sharing vector. Therefore, we believe it highly likely that many-core CMPs will employ CV directories to reduce overhead. Overhead per cache line drops to a mere 3.125% when the directory is configured as $Dir_2CV_{16}$. A directory with four pointers has twice the overhead (6.25%). This savings in storage overhead can substantially increase latency and bandwidth demands.

**Latency Impact.** Interconnect latency can be informally characterized by the following equation:

$$T = H \cdot T_{link} + H \cdot T_{router} + T_c + T_s$$

where $H$ is the average hop-count, $T_{link}$ the latency to traverse the wiring between routers, $T_{router}$ the delay a packet experiences through each router, $T_c$ the delay due to contention from other packets, and $T_s$ the serialization delay determined by the number of trailing flits. We target reducing the $H$ of invalidation requests and acknowledgments by truncating the NoC traversal of unnecessary invalidations and reducing the $T_c$ component of all packets by reducing the NoC load caused by extraneous invalidations and acknowledgments.

Fig. 1 shows the impact on average packet latency as the number of bits per directory is decreased for a $16 \times 16$ mesh with 4 different injection rates. Five percent of messages injected into the NoC are invalidation requests. The average number of destinations (sharers) per invalidate (prior to taking coarseness into account) is 2.5. All destinations are selected with a uniform random distribution. Table III (Section V) contains additional NoC parameters. All results are normalized to the latency measured for a full-map directory.

Load increases substantially[1] with a decrease in bits per directory entry; for each true sharer, $r - 1$ additional invalidation requests and acknowledgments may be generated. Even at moderate injection rates (10-15%), latency increases substantially. With 32 bits (8 cores/region) per directory entry, the NoC saturates at a 15% injection rate. With only 16 bits per directory entry, a $3.7x$ increase in latency occurs for a 10% injection rate and both 12% and 15% have saturated.

**Bandwidth Overheads.** CV directories increase bandwidth demands which has two implications for the NoC. First, this increase can drive up latency (and possibly saturate the NoC) as shown in Fig. 1. As NoC activity increases, router optimizations such as pipeline bypassing become less effective resulting in additional pipeline stages which drive up the average message latency and increases the $T_C$ component of packet latency. Second, this increase in activity will also results in more dynamic power consumption on the links.

[1]Injection rates are presented as the number of messages that would be injected into the NoC with a full-map directory; therefore with CVs, the actual injected traffic is higher than the stated injection rate.

TABLE I: Benchmark Descriptions

| Benchmark | Description |
|---|---|
| SPECjbb | Standard java server workload |
| SPECweb | Zeus Web Server 3.3.7 servicing 300 HTTP requests |
| TPC-W | TPC's Web e-commerce benchmark, DB Tier |
| TPC-H | TPC's Decision Support System running query 12 |
| Barnes | 8K particles, full end-to-end run including initialization |
| Ocean | 514x514 full end-to-end run (parallel phase only) |
| Radiosity | -room -batch -ae 5000 -en 0.050 -bf 0.10 (parallel phase) |
| Raytrace | car input (parallel phase only) |

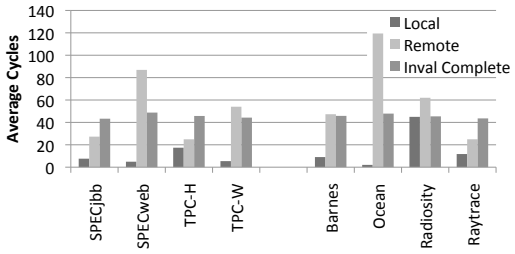


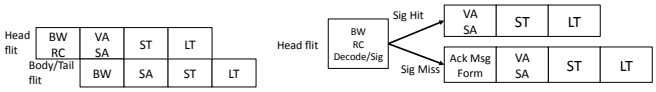

Fig. 2: Average Wait Time on Store Completion

**Impact of Invalidations.** In addition to increasing average packet latency, coarseness has a negative impact on invalidation latency. An upgrade cannot complete until all invalidation acknowledgments are received; this completion time increases with more sharers. The latency to complete invalidations can in turn have a negative impact on subsequent requests which may lead to significant overall performance degradation. To explore this potential impact, we examine the average delay that later requests experience due to a pending invalidation a full system simulator running 16 cores with a full-map directory. Table I provides workload descriptions. While a store to an address is outstanding, we measure the latency a subsequent cache access (either local or remote) to the same address must wait for the pending store to complete; the average is shown in Fig. 2. Additionally, we show the average latency for invalidates to complete (the requesting core has collected all acknowledgments). For 16 cores, the invalidation latency is low (on average 46 cycles); however, the use of coarse vectors in larger systems will increase invalidation delay causing these average wait times to grow substantially and degrade performance.

## III. SigNet: Invalidation Filter Architecture

SigNet uses cache summary information to remove extraneous invalidation messages that are sent when using CV directories. Counting Bloom filters are placed in each router to create signatures that summarize the cache contents of all tiles downstream (along a deterministic route) from the current router. A counting Bloom filter [10], [14] allows information to be removed from the signature; removal limits the likelihood that signatures will saturate and become ineffectual [2]. Absence of an address from the router's Bloom filter indicates that an invalidation request does not need to continue its NoC traversal as the destination cache block is invalid.

[2]Every signature lookup results in a hit indicating the *possible* presence of the address in a downstream cache.




(a) Baseline Router Pipeline (b) Modified SigNet Pipeline

Fig. 3: Router pipeline stages. BW: Buffer Write, RC: Routing Computation, VA: Virtual Channel Allocation, SA: Switch Allocation, ST: Switch Traversal, LT: Link Traversal, Decode: Decode Message Type, Sig: Signature Check.

The router signature filters targets $H$; messages are in-flight over shorter distances. $T_c$ component is reduced; lower $H$ decreases NoC load leading to less congestion.

To summarize the actions taken based on the signature data:

**Bloom Filter Hit:** A core exists along the dimension ordered (DOR) path between the current node and the destination that is caching an address mapping to the same entry: either the demand address or a false positive due to finite signatures.

**Bloom Filter Miss:** None of the caches downstream from this router are caching lines that map to this hash entry. Halting the invalidation at this node and generating an invalidation response is correct (a response must be generated since the directory will be receiving and counting acknowledgments; receiving too few acknowledgements will lead to deadlock).

**Bloom Filter Insertion:** Cache misses that travel to the directory increment the counter for the appropriate entry of the Bloom filter at each router along their traversal between the requesting core and the directory.

**Bloom Filter Deletion:** The writeback or invalidation acknowledgment associated with evicted cache blocks must decrement the Bloom filter counters between the cache and the directory. Invalidation acknowledgments are sent to the directory to ensure that the proper counters are decremented.

With the high level functions of SigNet filters in mind, we describe their integration into the router pipeline and microarchitecture.

### A. SigNet Architecture

Our baseline architecture (used to collect the data in Sec. II) is a canonical virtual channel router employing state-of-the-art optimizations to achieve single-cycle pipeline latency at low loads [16]. The baseline router pipeline is shown in Fig. 3a.

SigNet pipeline stage modifications (Fig. 3b) only apply to head flits; body/tail flits traverse the original pipeline (Fig. 3a). In the first stage, messages are decoded to determine the type. Messages may need to update/insert into the signature, check the signature or delete an item from the signature. Some messages such as data messages do not perform any signature operations and continue immediately to the VA/SA stage. Messages that only update the signature proceed to the VA/SA stage of the original pipeline.

Fig. 4 depicts SigNet's modified router architecture. Each message must be decoded to determine the message type; invalidation requests are handled differently from other message types. For an invalidation request, a signature lookup is performed on the message's desired output port; there is one signature per port in the router. The router uses lookahead routing; the signature lookup can be performed in parallel with the lookahead RC since the current output port is already known. A signature hit proceeds to the VA/SA (the top lane in Fig. 3b). Signature hits continue their NoC traversal because a downstream cache *may* be caching the block being invalidated. On a signature miss, the message arbitrates for the acknowledgment formation block which is another input/output port to the crossbar. Buffering is also associated with this block. Here the source and destination in the message header are swapped and the message type is switched to acknowledgment. Once the message is reformed, it proceeds to VA/SA (the bottom lane in Fig. 3b) so that it may return to the directory. Channel dependences are broken since the input message to acknowledgment formation block uses a different VC than the output message from the block.

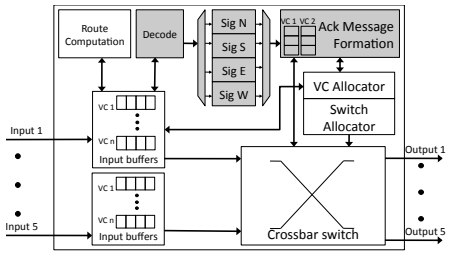



Fig. 4: Router Microarchitecture. Modifications shown in gray.

Some coherence messages must update the signatures. These updates are performed on the signature corresponding to the *input* port of the packet because the address being inserted or deleted is in a cache upstream from the current node. Requests to a new cache block will insert their address into the signature and eviction messages will decrement the appropriate signature counters.

The proposed router architecture employs class-based deterministic routing (CDR) [1]. CDR chooses to route a message along an XY or YX route depending on the message class. Here, the classes are request, reply and intervention. CDR allows requests to update signatures, while interventions traverse the reverse path to check the signatures and determine cache occupancy in downstream tiles.

### B. SigNet Example

Fig. 5 provides a walk-through example of SigNet. First, a cache miss to block A occurs and the request is sent to the directory (1). As Req. A traverses the NoC, it hashes A and inserts it into the signatures corresponding to its input ports (2). When Req. A reaches the directory (3), a bit is set in the coarse vector corresponding to the requester (assuming the directory entry is operating in overflow mode). Later, another core wants to upgrade its copy of A so it can perform a store to A (4). This upgrade request is sent to the directory. The directory forwards the invalidations to cores W, X, Y and Z; however of those 4 cores, only W is caching a copy of A. The invalidations destined for W and X hit in the filter indicating that a downstream cache may be caching A (6). Invalidations to Y and Z will take the west port and experience a filter miss indicating no downstream copies of A.

The directory waits and collects all acknowledgments and then sends a single combined acknowledgment to the requesting cache. When the invalidations destined for Y and Z experience a filter miss and are squashed, they must each individually arbitrate for the ack formation block. One acknowledgment each will be returned to the directory (7). Once the invalidation requests have reached W and X, invalidation acknowledgments will be sent back along an XY path to the directory.

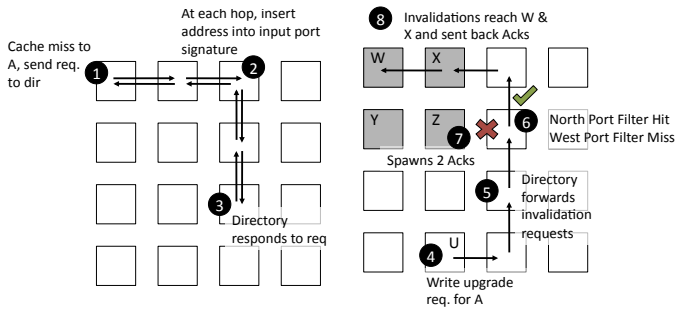



Fig. 5: Invalidation Filter Request/Acknowledgment Example

TABLE II: Per cache line overheads of CV + signatures.

| **Full Map** | $Dir_2CV_{16}$ | $Dir_4CV_8$ |
|---|---|---|
| No Signatures | | |
| 50% (32B/entry) | 3.125% (2B/entry) | 6.25% (4B/entry) |
| 8192 entry signatures with 6 bit counters | | |
| 50% (32B/entry) | 12.5% (8B/entry) | 15.625% (10B/entry) |
| 8192 entry signatures with 10 bit counters | | |
| 50% (32B/entry) | 18.75% (12B/entry) | 25% (14B/entry) |

The invalidation acknowledgment from X represents a *true* invalidation as X was caching the block in question. X's acknowledgment is tagged so the decode stage will cause X to decrement the appropriate signatures (to indicate that X is no longer caching this block) as it travels to the directory. Acknowledgments from W, Y, and Z were not from valid blocks and must not decrement the Bloom filters; the decode logic bypasses the update stage for these messages.

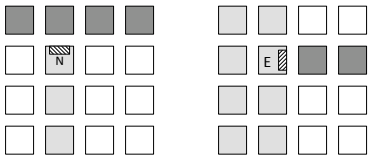
## IV. Implementation Issues

**Overhead.** SigNet adds storage overhead to the NoC; this overhead is substantially less than a full-map directory. SigNet places 8K entry counting Bloom filters at each output port of every router. The use of a counting Bloom filter allows entries to be deleted from the signature [10]. A Bloom filter entry counts the number of cache blocks downstream that map to this entry[3].

Table II shows the byte overheads of incorporating 8K entry signatures into the NoC. We show these costs as a function of additional bytes required per cache line. A full-map directory requires 32 bytes per cache line; the $Dir_2CV_{16}$ and $Dir_4CV_8$ require 2 and 4 bytes per line for directory pointers. Assume we have directory storage to map 64MB of data on-chip; the signatures are amortized across this area (signatures summarize many cache lines per entry) resulting in overheads of between 8 and 24 bytes per cache line for signatures plus the directory pointers. The additional logic to perform hashing and to update the signature is not included here. SigNet uses two XOR hash functions to hash into the Bloom filters. The address bits selected for the hash functions are taken from Notary [21].

**Correctness.** For correct execution, all cores caching a block being invalidated must receive the invalidation request. An invalidation request travelling from the directory to a core currently caching a request will not experience a filter miss during its NoC traversal. Deterministic routing provides this guarantee. Using CDR to send requests to the directory along XY path and having the directory send out responses and invalidations along an YX path will guarantee that if the line is cached downstream from the current node, then the address will hit in the filter at the current node (and every subsequent node until the destination is reached). The current SigNet architecture is limited to deterministic routing protocols; to facilitate adaptive routing protocols, more complex signature insertion (along multiple paths) would be required. Current NoCs favor simple deterministic routing schemes, so this is not a significant limitation.

[3]This can be multiple different addresses in one cache and across caches. The total can be $\#Addr\_mapped \times \#downstream\_cores$. Counters this large would be impractical; saturation is possible (and only has performance and not correctness concerns).




(a) North Port Example (b) East Port Example

Fig. 6: Cache summary information held in signatures

Bloom filters cannot have false negatives which would result in an incorrect execution. SigNet filters invalidations and spawns responses for in-flight messages; however it does not impact any ordering performed by the directory protocol. Support for task migration with SigNet is handled implicitly with the directory protocol; cache coherence requests would be required to move cache contents from the old core mapping to the new core mapping. This cache migration would update signatures appropriately.

Fig. 6 depicts two examples of cache contents that output ports must capture. In Fig. 6a, the north output port of Tile $N$ summarizes the subset of addresses in the caches of darkly shaded tiles that map to the directory home nodes of the light gray tiles. We assume directory information is uniformly distributed across all tiles; using a smaller number of directories would generate NoC hotspots but is larger orthogonal to the goals and implementation of SigNet. Fig. 6b, similarly shows the east output summary of Tile $E$. This signature summarizes all of the addresses in the two dark gray tiles that have as their directory home nodes any of the 8 light grey tiles. Different signatures will have different utilization by cache addresses; non-uniform signature sizes could be leveraged to improve the signature utilization, but is left for future work.

**False positives.** The rate of false positives in Bloom filters can limit the benefit of the SigNet architecture. However, even with a 100% false positive rate, the performance of SigNet will not degrade past the performance of the baseline CV directory.

Based on the analysis of Broder and Mitzenmacher [5], the probability of false positives in a Bloom filter is given by:

$$p = (1 - (1 - \frac{1}{s})^c)^h$$

where $s$ is the size of the Bloom filter, $c$ is the number of entries in each private cache and $h$ is the number of hash functions.

The Bloom filters in SigNet summarize the contents of multiple caches that lie between the directory and the destination of a message. Given the average hop count for a mesh NoC:

$$H_{avg} = \frac{nk}{3}$$

TABLE III: Configuration Parameters

| Network Configuration | |
|---|---|
| Number of Nodes | 256 |
| Topology | 16-ary 2-cube |
| Virtual Channels & Buffers | 4 VC/port, 8 Buffers/VC |
| Link Width | 16 Bytes |
| Signature Size | 8192 entry |

| Workload Parameters | | |
|---|---|---|
| **Name** | **% Invalidates** | **Avg Sharers** |
| Database | 6.0 | 2.3 |
| Web | 3.5 | 3.8 |
| Java | 2.7 | 2.2 |
| SciA | 2.0 | 2.3 |
| SciB | 5.0 | 3.0 |

which is $10\frac{2}{3}$ hops for a 16-ary 2-mesh. On average the partial contents of $\sim$11 caches will be summarized between the directory and the destination. Assuming that all cache accesses are uniformly distributed to directories across the 4 cardinal directions the probability of false positives in SigNet will be:

$$p = (1 - (1 - \frac{1}{s})^{\frac{nk}{3} \times \frac{c}{4}})^h.$$

With 32KB private caches per core, 8192 entries per bloom filter, 2 hash functions, the probability of false positives is just 2.5%. For larger private cache sizes, the probability increases: for 128KB cache, there is a 24.7% chance of false positives. Exploring topologies with lower hop counts will reduce the rate of false positives. For example, with a torus, the false positive probability in this scenario drops to 1.4% for a 32KB private cache size and to 15.5% for a 128KB private cache.

## V. Evaluation

In this section, we present the evaluation methodology followed by results indicating the performance and power potential of the SigNet architecture.

### A. Methodology

Limitations in current application scalability and in the overheads of full-system simulation preclude the study of large many-core architectures in such a manner. However, we use characteristics from 16-core workloads and extrapolate to create synthetic network workloads for larger systems to study the pressures CV directory entries place on the NoC.

We have characterized invalidations as a percentage of total messages using a full system simulator running 16 cores with 128KB of private cache per core. Additionally, we used this infrastructure to characterize the number of cores per invalidation. For the results in this paper, we assume that these characteristics will largely carry over to many-core systems.

Table III summarizes the NoC configuration parameters used for our evaluation and the parameters used to generate synthetic workloads for this section. As discussed earlier, several researchers have noted that the majority of cache blocks have a small number of sharers.

### B. Performance Results

We show average packet latencies for all 5 workloads with $Dir_2CV_{16}$ and $Dir_4CV_8$ configurations; a 10% injection rate is used. Results are normalized to a system with CV directories; there is a significant performance improvement with full-map directories. The addition of signature filters closes the gap between full-map and CV directories. With SigNet, latencies reductions of up to 28% are observed with an

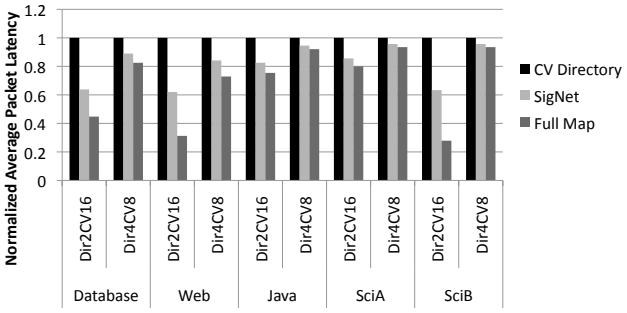



Fig. 7: Network Latency Results for SigNet

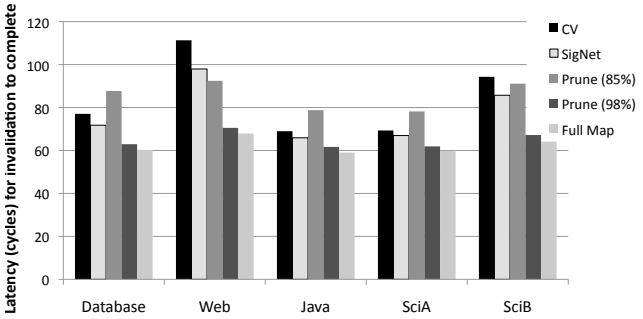



Fig. 8: Invalidation Completion Latency

average improvement of 20%. More dramatic improvements are observed with $Dir_2CV_{16}$ since more cores per region results in a larger number of injected invalidations in the NoC.

Fig 8 compares the total latency to complete an invalidation (sending the invalidation and receiving all acknowledgements). SigNet and CV results are shown for $Dir_4CV_8$. Pruning caches [19] with two different hit rates (98% and 85%) are included for comparison. Before traversing each dimension of the NoC, a pruning cache is checked to see if nodes in that dimension need to see the invalidation. Our NoC has 16 nodes in both dimensions; on a miss, the pruning cache will conservatively send invalidations to all nodes in the dimension. Here there are only two opportunities to prune; pruning caches are more effective with higher dimension topologies[4]. In general, for a low miss rate, pruning cache perform similarly to SigNet or slightly better. For a $Dir_2CV_{16}$, SigNet improves invalidation latency by 28% compared to a CV directory. A full cost comparison of building pruning caches that achieve these miss rates and SigNet is left for future work.

### C. Power Results

Next, we evaluate the potential dynamic power savings achieved with the SigNet architecture. Link traversal power and router power are calculated to be 0.397W and 0.739W respectively using Orion [12] and signature access power of 0.161W was taken from Notary [21]. Cache read power (tag only) is calculated to be 0.035W using Cacti [15]. Several assumptions are made to provide relative power consumption for each design. We assume 50% of messages access the cache and 70% of messages update or test signatures. The remaining 30% of messages are data transfers.

[4]Pruning caches also implement acknowledgment combining; this functionality is not currently evaluated; acknowledgment combining is orthogonal to SigNet and will be explored in future work.

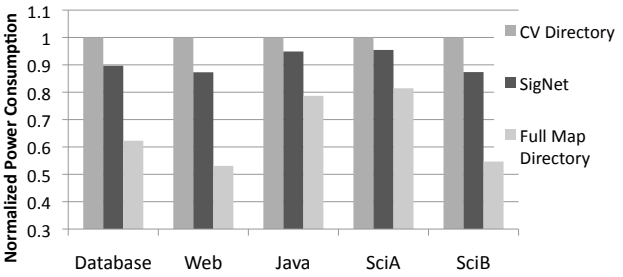



Fig. 9: Normalized Cache and Network Power

Figure 9 shows the power results (normalized to the $Dir_4CV_8$ configuration). SigNet provides significant savings by removing many hops from both invalidations and acknowledgments and prevents cache lookups from extraneous invalidations. SigNet saves an average of 21% (up to 25%) of power from cache lookups and network traversals over the baseline NoC with a $Dir_4CV_8$ directory. However, SigNet results in an average increase in NoC and cache power of $1.87x$ over the full-map directory. This comparison is pessimistic (for SigNet) as we do not account for the extra power consumption of the full-map directory over a CV directory.

We have compared SigNet against a full map and a CV directory; comparison of SigNet against other filtering proposals such as Jetty [14] is left for future work.

## VI. Related Work

In this section, we explore related research in coherence storage optimizations and co-design of interconnection networks and cache coherence protocols.

**Interconnection Network Support.** Pruning caches [19] are a means to filter invalidation requests and combine acknowledgments at various levels of the cache hierarchy. They note that these messages can cause interconnect and performance bottlenecks. Pruning caches only propagate necessary requests; they improve the scalability of both the protocol and the interconnect for large multiprocessor systems. INCF proposes in-network filtering support through region-based sharing information for snoop-based protocols [2].

**Cache Coherence Optimizations.** Characterization of broadcast systems has shown many snoop messages to be unnecessary [7]. Jetty [14] uses counting Bloom filters to represent the set of all blocks cached by a processor to reduce cache snoops. Blue Gene/P employs several filters to avoid unnecessary snoops [17]. Reducing the bandwidth of broadcast protocols improves the scability of bus-based systems. In this work, we examine eliminating unnecessary messages from the NoC to maintain the scalability of CV directory protocols.

LimitLESS [8] uses a limited number of pointers per directory information; when the number of sharers exceeds the number of pointers available, the processor traps to software to emulate full-map directory coherence. Tagless coherence directories [22] replace conventional directories with Bloom filters that store sharing information on a per-set basis to overcome the storage overheads of full-map directories.

**Signatures.** Signatures borrow from Bloom filters [4] and have been used in several proposals including transactional memory [20], [21]. Work by Sanchez et. al [18] focused on the implementation cost and overhead associated with signatures.

## VII. Conclusion

In this work, we characterize the impact of CV directories on the NoC power consumption and performance. Interconnect support for scalable coherence is imperative for many-core architectures. We present SigNet, a network architecture that reduces network load by filtering requests that do not need to be observed by the cores they are destined for. Extraneous network messages become quite pronounced as systems scale and force directories to leverage coarse vector mappings. The SigNet architecture improves average network latency by 20% and reduces both network power and cache power by 21% over the CV directory implementation.

## References

[1] D. Abts, N. Enright Jerger, J. Kim, D. Gibson, and M. Lipasti, "Achieving predictable performance through better memory controller placement in many-core CMPs," in *Proceedings of ISCA*, June 2009.
[2] N. Agarwal, L.-S. Peh, and N. K. Jha, "In-network coherence filters: Snoopy coherence without broadcasts," in *Proceedings of MICRO*, 2009.
[3] J. Archibald and J.-L. Baer, "An economical solution to the cache coherence problem," in *Proceedings of ISCA*, June 1985, pp. 355–362.
[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[5] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2005.
[6] H. Cain, K. Lepak, B. Schwarz, and M. H. Lipasti, "Precise and accurate processor simulation," in *Workshop on Computer Architecture Evaluation using Commercial Workloads*, 2002.
[7] J. Cantin, M. Lipasti, J. Smith, A. Moshovos, and B. Falsafi, "Coarse-grain coherence tracking: Regionscout and region coherence arrays," *IEEE Micro Top Picks*, no. 1, 2006.
[8] D. Chaiken, J. Kubiatowicz, and A. Agarwal, "LimitLESS directories: A scalable cache coherence scheme," in *ASPLOS*, 1991.
[9] J. H. Choi and K. H. Park, "Segment directory enhancing the limited directory cache coherence schemes," in *International Symposium on Parallel and Distributed Processing*, April 1999, pp. 258–267.
[10] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transaction on Networking*, vol. 8, no. 3, pp. 281–293, 2000.
[11] A. Gupta, W.-D. Weber, and T. Mowry, "Reducing memory and traffic requirements for scalable directory-based cache coherence schemes," in *International Conference on Parallel Processing*, 1990.
[12] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *Proceedings of Design and Test in Europe*, April 2009.
[13] M. M. K. Martin, P. J. Harper, D. J. Sorin, M. D. Hill, and D. A. Wood, "Using destination-set prediction to improve the latency/bandwidth tradeoff in shared memory multiprocessors," in *ISCA*, 2003.
[14] A. Moshovos, G. Memik, A. Choudhary, and B. Falsafi, "Jetty: Filtering snoops for reduced energy consumption in SMP servers," in *HPCA*, January 2001.
[15] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to understand large caches," Hewlett Packard, Tech. Rep., 2008.
[16] L.-S. Peh and W. Dally, "A delay model and speculative architecture for pipelined routers," in *Proceedings of HPCA*, January 2001, pp. 255–266.
[17] V. Salapura, M. Blumrich, and A. Gara, "Design and implementation of the Blue Gene/P snoop filter," in *Proceedings of HPCA*, February 2008.
[18] D. Sanchez, L. Yen, M. D. Hill, and K. Sankaralingam, "Implementing signatures for transactional memory," in *MICRO*, 2007.
[19] S. L. Scott and J. R. Goodman, "Performance of pruning-cache directories for large-scale multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 5, May 1993.
[20] L. Yen, J. Bobba, M. R. Marty, K. E. Moore, H. Volos, M. D. Hill, M. M. Swift, and D. A. Wood, "LogTM-SE: Decoupling hardware transactional memory from caches," in *HPCA*, February 2007.
[21] L. Yen, S. Draper, and M. D. Hill, "Notary: Hardware techniques to enhance signatures," in *Proceedings of the MICRO*, November 2008.
[22] J. Zebchuk, V. Srinivasan, M. Qureshi, and A. Moshovos, "A tagless coherence directory," in *Proceedings of MICRO*, 2009.