warm checkpoint is taken in the middle of the execution and with a reduced number of iteration runs. Transactional workloads are warmed up by running hundreds of thousands of transactions, and accurately simulated for a fixed number of transactions. SPEC workloads are fast-forwarded to the point of interest and simulate ~8billion instructions.

VI. Performance Results

A. Comparative results with reference protocols

The fundamental parameters of the system for the considered coherence protocols running the selected workloads, namely *execution time*, *average access-time* and memory hierarchy *energy delay product* (EDP) are shown from fig. 3 to fig. 5. To clarify the total storage, we denote it as SDE (*Sparse-Dir Equivalent*) capacity. All the results are normalized against token coherence and the directory size is being swept from a capacity to track 160% of the private cache blocks (8K SDE entries) to just 5% (32 SDE entries). Note that in order to keep implementation cost constant, in FLASK, at this point, half of this capacity is devoted to the filter and half to the directory. In other words, in the most extreme case, the FLASK directory will only have SDE capacity to track 16 shared blocks per controller (256 in the whole chip).

As expected, when the size of the directory is below one fourth of the aggregate private caches' capacity, the sparse-directory performance is degraded. In contrast to other previous works, [12], we observe this degradation with significant directory size reduction. This is related to the fact that in that work there is only one level of private caches. Here, L2 acts as a victim cache for L1. Thus, a directory induced private miss is eight times more frequent in L2 than L1. If we take into account that in L2, block reuse is low [18], the results seem reasonable. With an inclusive L1/L2 (which for an aggressive out-of-order core and a shared LLC might not be interesting), the effects will be more noticeable but still not too acute. Additionally, it should be noted that for an in-order core, directory-induced private misses (and subsequent hits in L3) will have more effect on performance. The memory level of parallelism present in the evaluated system allows the impact to be partially hidden.

As can be seen in fig. 5, sparse directory hits in private caches are reduced when the count of tracked blocks is decreased. Thus, there is a substantial increment as data must be

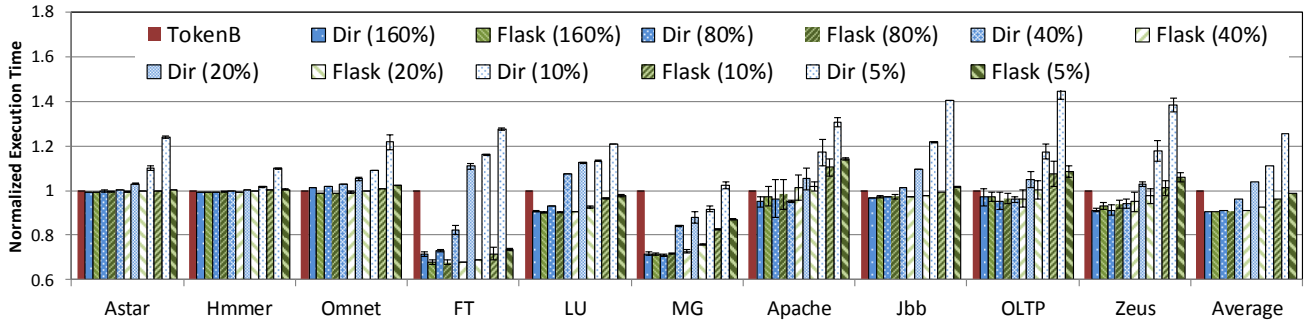

Fig. 3. TokenB Normalized Execution Time.

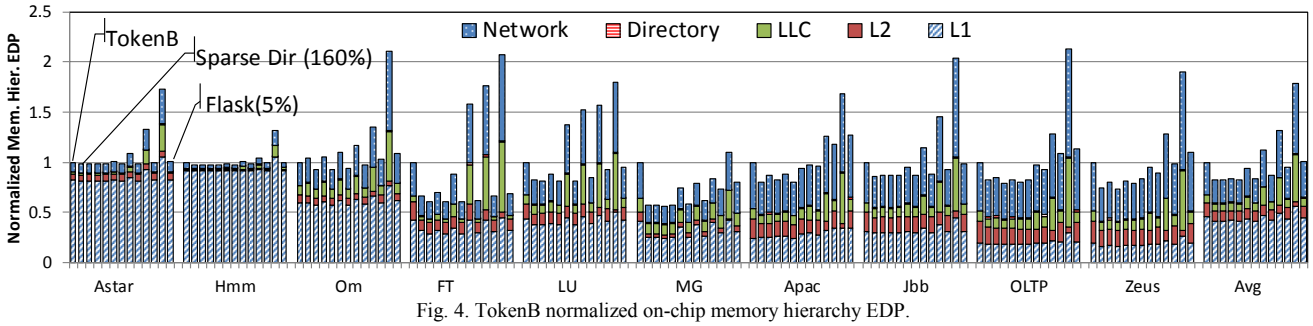

Fig. 4. TokenB normalized on-chip memory hierarchy EDP.

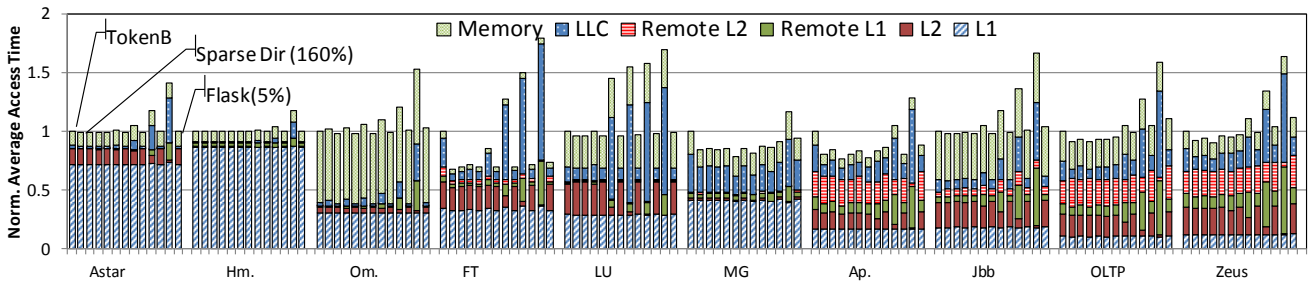

Fig. 5. TokenB normalized average memory access time.

retrieved from LLC (for private blocks) or other private caches (for actively shared blocks). In numerical or multi-programmed workloads the former case is more frequent while the latter exists in server workloads. This is consistent with the sharing degree. As can be seen for applications with a large portion of shared data, the latency degradation from 160% to 5% is almost doubled, degrading the performance by more than 40% on average. For these cases the intense coherence traffic due to directory-induced invalidations and subsequent LLC hits makes the activity increase in the network substantially, which degrades the energy properties, making it worse than the snoop-based protocol.

When we look at the performance of the snoop-based coherence protocol, we can observe unstable behavior. In some applications it seems to be almost the best performer while in others it is clearly not. The reason can be found in the on-chip contention. For some of the scientific applications, there is a significant traffic pressure on the LLC network. In spite of having a reasonably dimensioned network (4x4 mesh with 3 GHz clock and 16Byte links) and a router with state-of-the-art features [19] [23], the latency of the LLC is larger. Despite being an average sized system, extra traffic imposed by broadcast requests and the concurrent memory operations (at a given time there may be more than 16K packets in-flight) seem to surpass network capabilities. The solution for this unpredictable behavior is to oversize the network (e.g. increasing the link width, using topologies with better connectivity, using a more advanced router, etc.) or to redesign the coherence protocol [30]. Unfortunately this would be inappropriate for bigger systems. In any case, even in applications where TokenB has a slight performance advantage, the energy consumption is higher.

FLASK exhibits very different behavior to the other protocols. Even in the most extreme configuration, the performance seems quite unaffected. In the worst case, which is *apache*, the performance degradation observed versus token is 13%. In general, it seems that applications with low sharing degree are correctly handled, with almost negligible performance degradation. Having only SDE capacity to track 256 private blocks seems to affect other applications more significantly. This causes more reconstructions, which delays access to shared data after a capacity miss in private caches, slightly lengthening LLC access time. In other applications, such as the multi-programmed workloads, in spite of having a minimal filter (just 32 buckets per sub-table), the effects both on performance and energy of this extreme configuration are negligible. On average the performance degradation when reducing from 160% SDE capacity to 5% is only 8%, which is a noteworthy result. In all cases, the number of private-cache external invalidations due to the replacement of blocks in LLC without all the tokens is negligible.

Since the on-chip traffic is filtered out, there is no similar behavior to that observed for TokenB. In general, the energy requirements of the protocol are smaller than token and are quite steady regardless of the directory size. Note that this metric is pessimistic, since we exclude the cores' power consumption. As FLASK is the best performer in most cases, the EDP of these components will be low.

In summary, with almost no tracking capability, FLASK is more stable and energy efficient than broadcast-based protocol, and its performance degrades much more gracefully than conventional sparse-directory when directory size is reduced.

B. *Filter Efficiency*

The previous results contain some details that we want to highlight, namely, the filter efficiency. The main figure of merit is the false positives, which increase on-chip energy and memory

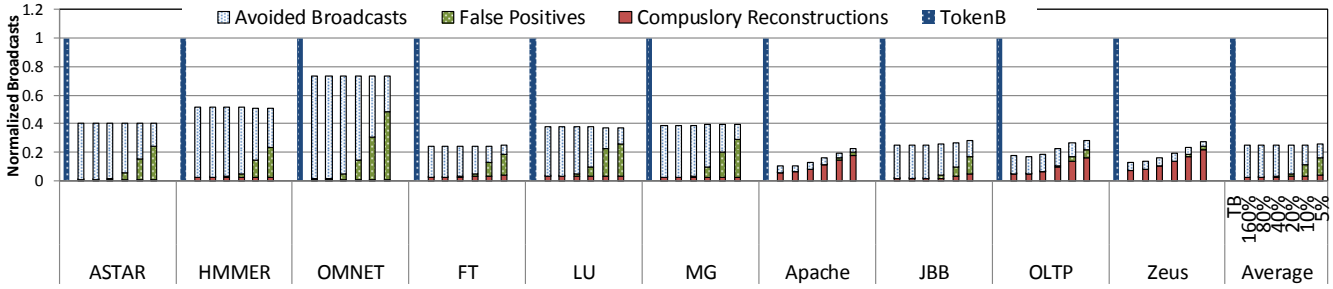

Fig. 6. TokenB normalized filter efficiency for different SDE sizes.

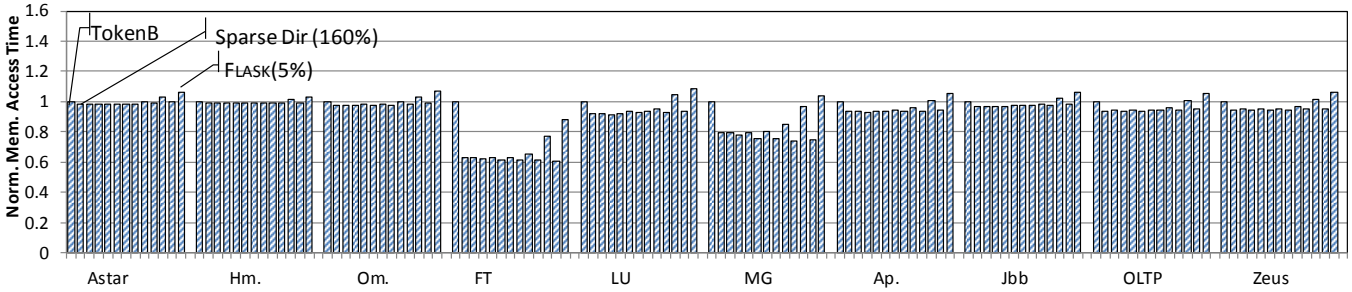

Fig. 7. TokenB normalized memory latency overhead.

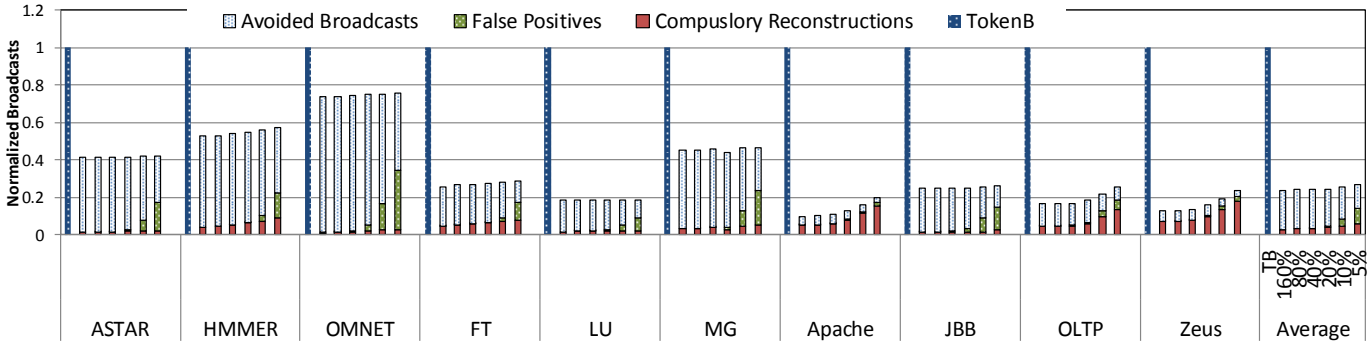

Fig. 8. TokenB normalized traffic filtering with adaptive partitioning.

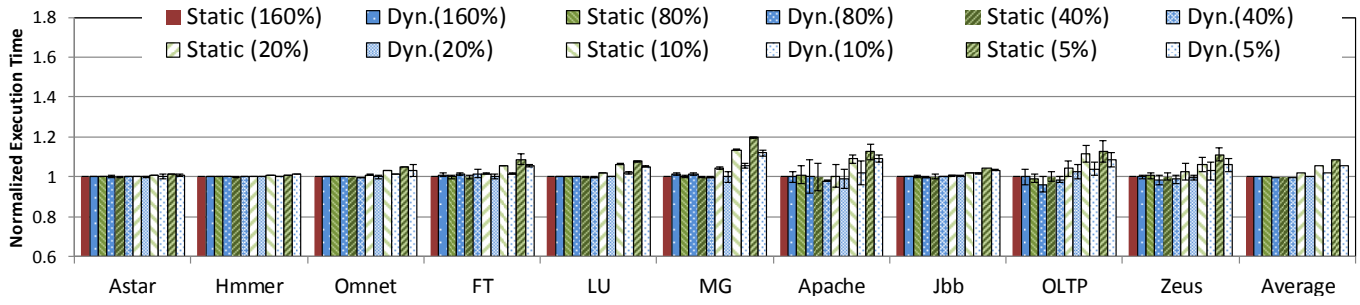

Fig. 9. 40% SDE Static normalized performance of FLASK with dynamic and static storage allocation.

access time. Fig. 6 shows the total number of multicasts over a range of available storage for the filter capacity.

For reference, we normalize this metric to multicast in TokenB. Note that, when we take into account that the network can natively handle multicast traffic and the cache snoop energy contribution, this might not be directly transferred to link utilization or energy consumption, as we can contrast these results with fig. 4. When the directory is dimensioned for 20% of private caches, false positives seem to be consistent with the theoretically expected proportion of 5%. If we shrink the SDE capacity to just 5% of private caches, although there is an increase in false positives, counter saturation never occurs in any of the runs of our evaluations. While in some cases, even with such a small filter, it is able to detect a reasonable number of on-chip misses; in other cases (such as the NAS applications), it cannot do so. In some applications, the number of true positives (private cache misses for actively shared data) grows when we reduce the size of the directory while in others, it remains almost unchanged. In the former, the directory is not able to maintain the shared working set, whereas in others it can do so. In contrast to this behavior, false positives are more numerous in the latter applications as private blocks increase the number of elements to be tracked by the filter.

Fig. 7 shows the TokenB normalized memory access time (from the core perspective, i.e. includes the whole latency to transmit the desired word from memory to the processor back-end) for FLASK and sparse-dir. Note that memory requests in both cases will be roughly the same, since most off-chip requests are induced by LLC capacity misses and both protocols handle LLC data in the same way. In numerical applications, false positives for very small filter capacities have a negative effect due to the delay in off-chip access and the on-chip latency that the extra traffic adds. Consequently, there is a significant increase in memory access time. In applications such as the server workloads, the extra traffic due to compulsory reconstructions increases the contention, increasing the access time to the memory controller and therefore delaying the memory access.

For the generous off-chip bandwidth considered, the results are dominated by on-chip effects: on-chip contention and the additional latency induced by false positives (which delays the memory request until the coherence controller realizes that there is no on-chip copy). Even in the most adverse directory configuration, the effect is less than 5%. This is almost unnoticeable in the average access time, as can be appreciated in fig. 5. Note that in a system with many cores or large private caches, this configuration could reduce the storage footprint significantly. In any case, directory size can be a useful knob for the trade-off between implementation and energy cost.

C. *Adaptive Filter/Directory resource partitioning*

As stated before, dlCBF allows adaptive resource splitting. Although we did not implement the on-line adaptation, we can statically choose the best configuration for the application. Looking at fig. 6, it seems clear that multi-programmed, numerical applications require few entries in the directory. On the contrary, for commercial workloads, the on-chip traffic is dominated by reconstructions. Therefore, we should select only a 1-way directory for the first two classes (with 7 sub-tables in the filter) and a 7-way directory for the last class (with just 1 sub-table in the filter). Note that even in a multi-programmed workload, a small number of OS addresses might be shared, so we need at least a 1-way directory. The broadcast signature, normalized against the token, is shown in fig. 8. The

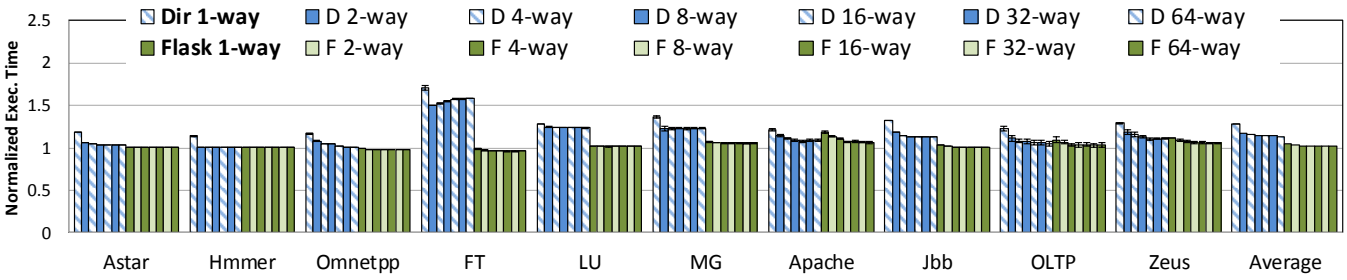

Fig. 10. 160% SDE sparse directory normalized performance for different associativities (20% SDE).

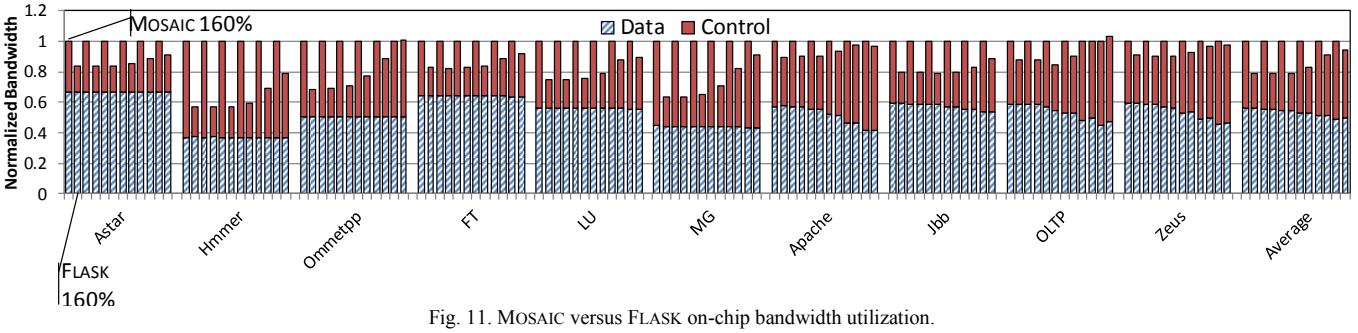

Fig. 11. MOSAIC versus FLASK on-chip bandwidth utilization.

performance results of the static and dynamic storage allocation in the memory controller are shown in fig. 9. If we compare these results with those provided in fig. 6, the effect of the approach is to halve the storage resources with negligible impact. For example, under these conditions with 10% SDE capacity, the traffic requirements are close to 20% of SDE capacity when the results are statically halved. Consequently with just 10%, in most applications, a tolerable false positive rate is observed. On average, with 10% and 5% of SDE capacity, we observe a performance penalty of about 3% and 6% respectively. Although dlCBF is suitable for an on-line mechanism, capable of morphing sparse-directory and filter throughout the application execution, we left this analysis open for future work. Under cloud-computing scenarios (with live workload migration), this might be interesting, given the results seen here. However, in any case, we believe that such a task should be done in the software layer, since the switch in behavior will be quite infrequent.

D. Comparing FLASK with other directory cost reduction alternatives

FLASK can be combined with other strategies focused on the same goal [11][12][13][27][32]. Some of them are focused on eliminating the directory overprovision by emulating large associativity through multi-hashing indexing and insertion [32] [33], achieving in most workloads the benefits of a very high associativity at a fraction of the cost. Therefore, if we increase the associativity of the sparse directory, we can achieve a similar effect. Fig. 10 shows the over- provisioned normalized result for different associativities (ranging from 1-way to 64-way with a fixed capacity). In order to appreciate conflict misses in the directory, and given that the computational requirements of the evaluation framework system and application scalability prohibit it, we artificially reduce the size to just 20% SDE (otherwise evictions caused by conflict in the directory are negligible). Except in the case of FT, whose counterintuitive behavior is caused by very low directory reuse [32], increasing the associativity reduces the directory conflicts, which provides a slight performance degradation. As we can appreciate, 1-way directory in FLASK is able to outperform the sparse directory even with 64 ways. In FLASK, the impact of directory conflicts on performance is negligible. Therefore, FLASK will provide better performance than techniques focused on minimizing directory conflicts such as [11][12]. The reasons for this are: (1) there is no need to perform external invalidations after a directory eviction, and (2) it only uses the directory to track actively shared blocks. Under such circumstances, the experimental observations made by [15] about conflicts are no longer applicable. In fact, although not exploited here, FLASK can be used to reduce directory implementation cost (v.gr. using very low associativity).

Like FLASK, MOSAIC was focused on improving directory scalability, eluding directory inclusiveness through entry reconstruction on demand. Fig. 11 compares the on-chip memory bandwidth consumption for the two approaches, for different SDE capacities. As expected, the control traffic generated by FLASK is significantly less than MOSAIC. This is because directory entry reconstructions in FLASK are performed only for shared data. This means that in some multi-programmed workloads, such as *hmmer,* the total amount of traffic is up to 60% less. When the size of the filter is reduced, the false positives increase the traffic slightly. With applications with a high sharing degree, such as *apache,* this advantage is smaller. In this last type of benchmarks, when the size of directory is reduced below 20%, the behavior of the two protocols becomes more similar due to the fact that most reconstructions are because of shared blocks. On average, and with a directory of at least 20%, FLASK enables the reduction of total on-chip traffic by 20%.

VII. CONCLUSIONS

We have proposed an evolutionary re-architecture for directory coherence protocols that can benefit from snoop-based coherence without paying a high toll. The results enable a balanced approach that improves system performance in a wide range of applications.

The proposal might be beneficial to scale the coherence protocol for many-core systems or for medium-size CMPs, as we have demonstrated in the results section especially so bearing in mind recent and forthcoming commercial systems. In any case, we have shown that even for 16-core CMPs there are both power and performance benefits versus other protocols.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their helpful comments. Special thanks to Dan Sorin and Viji Srinivasan for their valuable suggestions.

REFERENCES

[1] P. Abad et al., "TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers," in *Int. Symposium on Networks-on-Chip (NOCS)*, 2012, pp. 99–106.

[2] A. R. Alameldeen et al., "Simulating a $2M commercial server on a $2K PC," *Computer (Long. Beach. Calif).*, vol. 36, no. 2, pp. 50–57, Feb. 2003.

[3] M. Alisafaee, "Spatiotemporal Coherence Tracking," in *Int. Symposium on Microarchitecture (MICRO)*, 2012, pp. 341–350.

[4] J. Almeida and A. Z. Broder, "Summary cache: a scalable wide-area Web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Jun. 2000.

[5] I. Atta, P. Tözün, X. Tong, A. Ailamaki, and A. Moshovos, "STREX," in *Int. Symp. on Computer Architecture (ISCA)*, 2013, vol. 41, no. 3, p. 273.

[6] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[7] F. Bonomi, M. Mitzenmacher, and R. Panigrahy, "An improved construction for counting bloom filters," in *ESA'06 14th Annual European Symposium*, 2006, pp. 684–695.

[8] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, Jan. 2004.

[9] M. Butler, "AMD 'Bulldozer' Core - a new approach to multithreaded compute performance for maximum efficiency and throughput," in *Symposium on High-Performance Chips (HotChips)*, 2010.

[10] P. Conway, N. Kalyanasundharam, G. Donley, K. Lepak, and B. Hughes, "Cache Hierarchy and Memory Subsystem of the AMD Opteron Processor," *IEEE Micro*, vol. 30, no. 2, pp. 16–29, Mar. 2010.

[11] B. A. Cuesta, A. Ros, M. E. Gómez, A. Robles, and J. F. Duato, "Increasing the effectiveness of directory caches by deactivating coherence for private memory blocks," in *Int. Symp. on Computer Architecture (ISCA)*, 2011, p. 93.

[12] S. Demetriades and S. Cho, "Stash Directory: A Scalable Directory for Many-Core Coherence," in *International Symposium on High Performance Computer Architecture (HPCA)*, 2014.

[13] M. Ferdman, P. Lotfi-Kamran, K. Balet, and B. Falsafi, "Cuckoo directory: A scalable directory for many-core systems," in *2011 IEEE 17th Int. Symp. on High Performance Computer Architecture*, 2011, pp. 169–180.

[14] E. J. Fluhr et al., "POWER8: A 12-core server-class processor in 22nm SOI with 7.6Tb/s off-chip bandwidth," in *International Solid-State Circuits Conference (ISSCC)*, 2014, pp. 96–97.

[15] A. Gupta, W. Weber, and T. Mowry, "Reducing memory and traffic requirements for scalable directory-based cache coherence schemes," in *Int. Conference on Parallel Processing (ICPP)*, 1990, pp. 312–321.

[16] P. Hammarlund et al., "Haswell: The Fourth-Generation Intel Core Processor," *IEEE Micro*, vol. 34, no. 2, pp. 6–20, 2014.

[17] J. Huh et al., "A NUCA substrate for flexible CMP cache sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 8, pp. 1028–1040, 2007.

[18] A. Jaleel, K. B. Theobald, S. C. Steely, and J. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)," in *Int. Symposium on Computer Architecture (ISCA)*, 2010, pp. 60–72.

[19] N. E. Jerger, L. S. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *International Symposium on Computer Architecture (ISCA)*, 2008, pp. 229–240.

[20] H. Jin, M. Frumkin, and J. Yan, "The OpenMP implementation of NAS parallel benchmarks and its performance," *NASA Ames Research Center," Technical Report NAS-99-011*, Citeseer, 1999.

[21] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-Way Multithreaded Sparc Processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, Mar. 2005.

[22] T. Krishna, L.-S. Peh, B. M. Beckmann, and S. K. Reinhardt, "Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication," in *Int. Symposium on Microarchitecture (MICRO)*, 2011, vol. 2, pp. 71–80.

[23] A. Kumary, P. Kunduz, A. P. Singhx, L.-S. Pehy, and N. K. Jhay, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *International Conference on Computer Design (CCD)*, 2007, pp. 63–70.

[24] M. M. K. Martin, M. D. Hill, and D. A. Wood, "Token Coherence: decoupling performance and correctness," in *30th Annual International Symposium on Computer Architecture, 2003. Proceedings.*, 2003, pp. 182–193.

[25] M. M. K. Martin, D. J. Sorin, M. D. Hill, and D. A. Wood, "Bandwidth adaptive snooping," in *International Symposium on High Performance Computer Architecture (HPCA)*, pp. 251–262.

[26] M. M. K. Martin et al., "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *ACM SIGARCH Comput. Archit. News*, vol. 33, no. 4, p. 92, Nov. 2005.

[27] L. G. Menezo, V. Puente, and J. A. Gregorio, "The case for a scalable coherence protocol for complex on-chip cache hierarchies in many-core systems," in *Int. Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2013, pp. 279–288.

[28] A. Moshovos, "RegionScout: Exploiting Coarse Grain Sharing in Snoop-Based Coherence," in *Int. Symp. on Computer Architecture (ISCA)*, 2005, pp. 234–245.

[29] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *Int. Symposium on Microarchitecture (MICRO)*, 2007, pp. 3–14.

[30] A. Raghavan, C. Blundell, and M. M. K. Martin, "Token tenure: PATCHing token counting using directory-based cache coherence," in *Int. Symposium on Microarchitecture (MICRO)*, 2008, pp. 47–58.

[31] M. V. Ramakrishna, E. Fu, and E. Bahcekapili, "Efficient hardware hashing functions for high performance computers," *IEEE Trans. Comput.*, vol. 46, no. 12, pp. 1378–1381, 1997.

[32] D. Sanchez and C. Kozyrakis, "SCD: A scalable coherence directory with flexible sharer set encoding," in *18th IEEE Int. Symposium on High Performance Computer Architecture*, 2012, pp. 1–12.

[33] D. Sanchez and C. Kozyrakis, "The ZCache: Decoupling Ways and Associativity," in *MIcro 2010*, 2010, pp. 187–198.

[34] D. Sanchez, L. Yen, M. D. Hill, and K. Sankaralingam, "Implementing Signatures for Transactional Memory," *Int. Symp. Microarchitecture*, pp. 123–133, 2007.

[35] SPEC Standard Performance Evaluation Corporation, "SPEC 2006." .

[36] C. Sun et al., "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in *Int. Symposium on Networks-on-Chip (NOCS)*, 2012, pp. 201–210.

[37] B. Vöcking, "How asymmetry helps load balancing," *J. ACM*, vol. 50, no. 4, pp. 568–589, Jul. 2003.

[38] "SLICC specification of Flask and Counterpart Coherence Protocols." [Online]. Available: http://www.atc.unican.es/galerna/flask.