Performance Evaluation of Exclusive Cache Hierarchies

Ying Zheng **Brian T. Davis** **Matthew Jordan**

Electrical & Computer Engineering Department,
Michigan Technological University
{yizheng, btdavis, mcjordan}@mtu.edu

Abstract

Memory hierarchy performance, specifically cache memory capacity, is a constraining factor in the performance of modern computers. This paper presents the results of two-level cache memory simulations and examines the impact of exclusive caching on system performance. Exclusive caching enables higher capacity with the same cache area by eliminating redundant copies. The experiments presented compare an exclusive cache hierarchy with an inclusive cache hierarchy utilizing similar L1 and L2 parameters. Experiments indicate that significant performance advantages can be gained for some benchmarks through the use of an exclusive organization. The performance differences are illustrated using the L2 cache misses and execution time metrics. The most significant improvement shown is a 16% reduction in execution time, with an average reduction of 8% for the smallest cache configuration tested. With equal size victim buffer and victim cache for exclusive and inclusive cache hierarchies respectively, some benchmarks show increased execution time for exclusive caches because a victim cache can reduce conflict misses significantly while a victim buffer can introduce worst-case penalties. Considering the inconsistent performance improvement, the increased complexity of an exclusive cache hierarchy needs to be justified based upon the specifics of the application and system.

1 INTRODUCTION

In modern computer development, the rate of improvement in microprocessor speed exceeds the rate of improvement in DRAM memory access latency. To solve this disparity, cache has been used to effectively diminish the impact of the cycle time differential. Cache memories are small, high-speed buffer memories, which contain the most recently used portions of main memory [1]. For on-chip cache, the cycle time of a small cache can match that of the processor, yielding an access time of 1-cycle. Cache hit rate for a small cache is limited. However, cache size cannot be increased without limitation. For a fixed technology, the larger the cache, the slower the cache will become [2], and larger caches increase the cost of manufacturing. Another way to increase cache hit rate is to more effectively use the existing cache area. Many desktop microprocessors use an on-chip two-level cache hierarchy. A two-level cache hierarchy allows a tradeoff between optimizing hit time and miss rate [9]. Two level caches can be designed to be either inclusive or exclusive. For example, the Intel Pentium® 4 Willamette [13] has an on-die 256kB inclusive L2 cache, while the AMD Athlon™ Thunderbird [14] has an on die 256kB exclusive L2 cache. An inclusive cache system implies that the contents of the L1 cache be a subset of the L2 cache [2]. This decreases the effective cache capacity available for unique information [3]. An exclusive cache hierarchy is a hierarchy in which the contents of the L1 and L2 are exclusive. This means that the effective cache size is L1+L2, unlike inclusive caching, whose effective cache size is the size of the largest cache – typically the L2.

Using an exclusive cache hierarchy, with the same die area dedicated to storage as a traditional inclusive cache, larger effective cache size can be obtained. The exclusive cache state machine is more complex, and an exclusive cache requires a victim buffer. These factors may result in a slightly larger combined circuit for the exclusive cache, but for a large cache the transistors allocated to storage dominate the circuit area required. In the studies presented, inclusive and exclusive caches of equal size, and with victim caches or victim buffers respectively of equal size, are compared.

2 BACKGROUND

Exclusive caching is one technique used to maximize the effective cache size. Jouppi and Wilton evaluated a direct-mapped and 4-way set-associative exclusive cache compared with a conventional inclusive cache. The results showed that the exclusive cache has improved performance over inclusive cache due to the increased on-chip capacity [5]. The concept of an exclusive cache hierarchy can be relevant to any microprocessor implementation utilizing multiple levels of cache. Theodore et.al. examined the usage of exclusive caching in networking storage devices

This work is supported by NSF CAREER Award CCR – 0133777.

[4]. In this study, only on chip microprocessor cache performance is addressed.

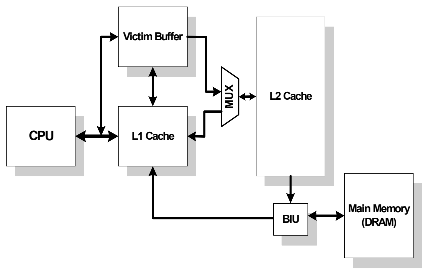


Figure 1: Exclusive Cache Architecture

AMD introduced an implementation of an on-die two-level exclusive cache architecture in the Athlon™ and Duron™ processors in 2000 [3]. Figure 1 illustrates the architecture. The L2 cache contains only victim or copy-back cache blocks that are ejected from the L1 due to conflict misses. When a miss occurs in the L1, the victim block is transferred to the victim buffer (VB) while at the same time the L2 is checked. There is 128KB of L1 cache, of which 64KB is for instruction and 64KB is for data. This provides a hit rate sufficiently high to ensure that the victim buffer is rarely fully occupied; the microprocessor can flush the victim block to the L2 when it is idle some time after the load request has been serviced, hiding the traffic. The scenario which results in the highest L2 latency for the exclusive cache architecture occurs when the victim buffer is full and there is a L1 miss/L2 hit. In this worst case scenario a three-step sequence is required. The victim buffer entry must first be flushed to the L2, after which the L1 victim is moved to the victim buffer. Only then can the requested data be retrieved from the L2.

The victim buffers play a different role than that of a victim cache, although both are used to hold victims ejected from an L1 cache. To minimize the stalls imposed upon the processor by the requirement of an available victim buffer entry to service an L1 miss, the victim buffer flushes the data as soon as the bus is idle. This is in contrast to a victim cache, which maintains data all blocks in the hope that the data is reused soon. To maintain coherence there are cases in an exclusive cache hierarchy when the victim buffer must function as a victim cache. This occurs when written data has not been flushed completely from the victim buffer and the data requested by the processor is contained in the victim buffer. When this happens, the victim buffer swaps the data with the block in the L1.

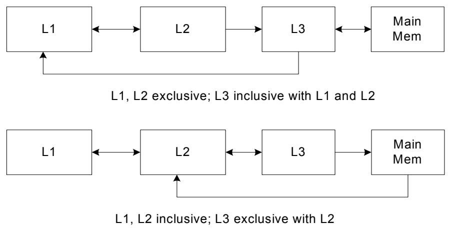

L1
L2
L3
Main Mem
L1, L2 exclusive; L3 inclusive with L1 and L2
L1
L2
L3
Main Mem
L1, L2 inclusive; L3 exclusive with L2

Figure 2: Three-level exclusive cache

Exclusive caching or inclusive caching can be extended to multiple levels if more than two levels of cache are present. A multi-level exclusive hierarchy could be designed in many ways, and could include three-levels of cache with L1 and L2 inclusive and L3 exclusive, or L1 and L2 exclusive and L3 inclusive. Figure 2 shows these two architectures.

Movement of blocks between cache levels is made easier through the use of a common line size. Unequal cache line size will result in a cache hierarchy that is potentially only partially inclusive. Fragmentary exclusivity may occur when L1 is set associative, the block size is not the same for L1 and L2, or the number of sets in L2 is smaller than that of L1 [8]. Fragmentary exclusivity implies that some but not all of the L1 contents are duplicated in L2, so that the effective cache is less than L1+L2.

Many multiprocessors use a multilevel cache hierarchy to reduce both the demand on global interconnect and the cache miss penalty. Inclusive caching simplifies the well-understood cache coherence problem for multiprocessors. Some multiprocessors share address space, and for such systems, a given piece of information may have several copies in different places, which can cause coherence problems. Many algorithms that are used to maintain memory consistency enforce inclusive caching. If exclusive caching is used in a multiprocessor environment, then the L1 cache needs dedicated snoop ports as used in the AMD Athlon MP processor [11]. This method solves the coherence problem, but at the cost of die area and cache access latency.

In this paper, two-level exclusive cache hierarchies in a uniprocessor system are compared with two-level inclusive cache hierarchies of equivalent size. The victim buffer, which is necessary for an exclusive cache hierarchy, is the most significant resource differential between the exclusive and inclusive caches compared. In an attempt to minimize the differences between the inclusive and exclusive hierarchies, each inclusive hierarchy simulated has a victim cache comparable to the victim buffer required by the exclusive cache of the same size. The cache line size is fixed across L1, L2, and VC/VB for each configuration. The L2 has only one port, which is common practice in most modern microprocessors, since a single ported cache requires less area than a two-ported cache. The victim buffer or victim cache has a variable number of entries, and it shares the bus with the L1 as shown in Figure 1.l

3 SIMULATION TECHNIQUES

To evaluate the performance of the exclusive cache system, a simulator for a parameterized exclusive cache hierarchy was built. This simulator was integrated into sim-outorder from the SimpleScalar toolset version 3.0c [6].

3.1 Experimental Settings

Without modification, sim-outorder models a two-level cache system, which uses a write back policy for the L1 cache and a write through policy for the L2 cache using a write allocate strategy. For the studies presented herein, this toolset is expanded to include three new models, which are integrated into the simulation environment. These three models are: an exclusive cache model, a victim buffer model and a DRAM (Dynamic Random Access Memories) model, as shown in Figure 1. Each model is modular and parameterized, and can be optionally incorporated in each simulation.

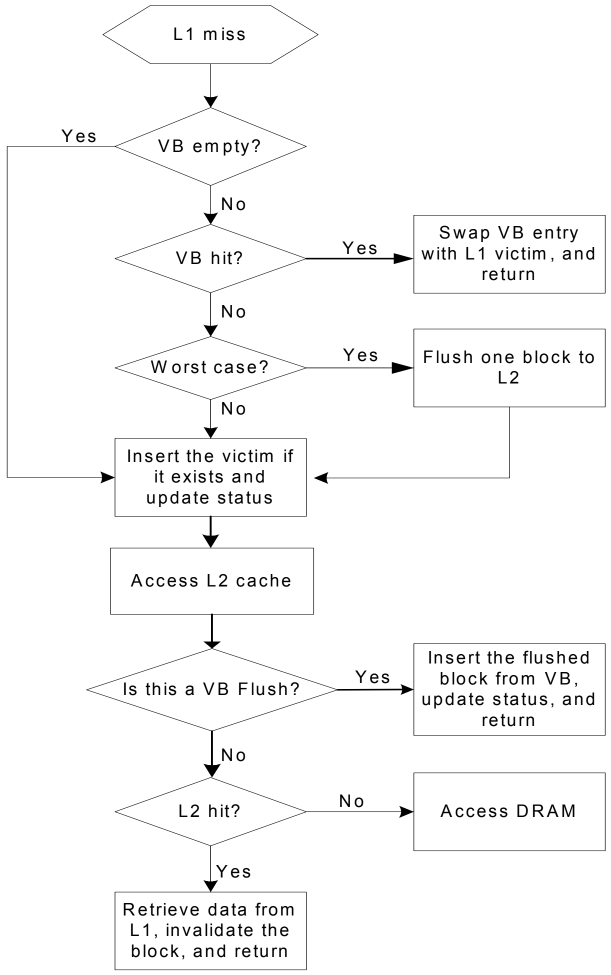
3.2 Model Description

Inclusive caching has been the standard cache behavior due to its simplicity. For an inclusive cache, when a load misses in the L1 but hits in the L2, the cache block is copied from the L2 to the L1. If the load misses in both the L1 and L2 cache, the data is retrieved from the main memory to the L2 cache, which then provides the data to the both the L1 and L2 caches each of which allocate a redundant copy

In two-level exclusive caching, when a load misses in the L1 and hits in the L2, the contents of L1 and L2 are swapped. That is, the victim block from the L1 cache is first transferred to the victim buffer; then the referenced data is transferred from the L2 cache into the L1 cache, after the request is serviced, the victim block is transferred to the second-level cache. If a load misses in the both the L1 and L2, the desired item is copied directly into the L1 from the main memory, potentially evicting a victim to the L2 cache. Under this scheme, data cannot be in both L1 and L2, which gives rise to “exclusion”.

Whenever a cache is exclusive, a victim buffer is necessary to allow a read to be serviced prior to write completion. The victim buffer modeled between L1 and L2 utilizes a first-in first-out (FIFO) replacement policy. To make the assessment comparable, the inclusive cache simulated also includes victim cache with the same size, which is accessed in parallel with the L1 cache.

Figure 3 is a flow chart of the finite state machine (FSM) controlling accesses to a simulated exclusive two-level cache. When the CPU performs a memory access, it will check the L1 cache and the victim buffer simultaneously, and either one can service the request if the data is available. It should never be the case that the L1 cache and the VB contain the same data element since they are also exclusive. Load instructions have priority to access the L2 while the victim buffer contains write data destined for the L2 cache; the victim buffer is designed to flush data only when the bus is idle. When the worst-case scenario occurs, the victim buffer drains only a single cache element, so as to minimize CPU stalling time, and allow the following access to be serviced as soon as a VB entry is available. Another potential advantage of an exclusive cache hierarchy is that the victim buffer can hold data longer in the same manner as a victim cache, allowing increased performance if a hit in the victim buffer occurs in the near future.



L1 miss
VB empty?
Yes
No
VB hit?
Yes
Swap VB entry with L1 victim, and return
No
Worst case?
Yes
Flush one block to L2
No
Insert the victim if it exists and update status
Access L2 cache
Is this a VB Flush?
Yes
Insert the flushed block from VB, update status, and return
No
L2 hit?
No
Access DRAM
Yes
Retrieve data from L1, invalidate the block, and return

Figure 3: Block Diagram Of The Exclusive Cache

The FSM logic of an exclusive cache is more complex than conventional inclusive cache, but more significantly, it integrates the control of the multiple on-chip caches.

For both inclusive caching and exclusive caching, if the request is missed in the L2, the request goes on to the main memory. In SimpleScalar 3.0c the main memory access latency is fixed to be 32 cycles with infinite parallelism. In order to improve the accuracy of this main memory model, a DRAM model is incorporated to provide a more realistic latency based upon the non-uniform access latency of real DRAM. In the new DRAM model, the memory controller and a bus with contention are emulated. Bank contention, DRAM precharge, and DRAM refresh are also considered, and several DRAM timing configurations can be selected.

The exclusive L2 cache model verification was performed through deterministic benchmarks, the procedure of which is available [15].

3.3 Methodology and Benchmarks

10 benchmarks from SPEC2000 [7] were used in the experiments presented. These benchmarks were precompiled binaries targeted for the PISA_SS_Little ISA and were randomly selected. Each benchmark was simulated using a modified SimpleScalar 3.0c on RedHat Linux machines. Table 1 lists the benchmarks used in this study. Also included is the input set used for the benchmark and the number of load instructions simulated for the benchmark.

Table 1: Benchmarks for Simulations

Name	Input	Number of Load Instructions
ammp00	ammp	1.64E+09
art00	test	4.11E+08
bzip200	test	3.68E+08
equake00	inp	1.85E+09
gcc00	cccp	4.14E+08
gzip00	test	5.87E+08
mcf00	inp	40628928
mesa00	test	5.55E+08
vortex00	lendian	2.79E+09
vpr00	test	2.07E+08

To accurately model the non-uniform latency of a synchronous DRAM memory system a DDR333 / PC2700 SDRAM model is integrated into sim-outorder. The timing characteristics of the DRAM T_{cl}-T_{rcd}-T_{rp} are 2-2-2, and the bus multiplier is 12, implying a processor clock speed of 2GHz. SimpleScalar 3.0c assumes that a write buffer of unlimited size is used, reads are allowed to bypass writes, and the buffered data is written to main memory whenever there are no outstanding reads. Writing from the victim buffer to the L2 cache is pipelined, and the bus between the victim buffer and the L2 cache is assumed to be as wide as the cache line size, so that the latency for writing each entry is one-cycle.

4 SIMULATION RESULTS

The simulations compare the performance of an inclusive cache with an exclusive cache, where the cache configurations vary in: L2 size alone, L1 and L2 cache size, and the number of victim buffer or victim cache entries. The number of L2 cache accesses, L2 misses, execution time and other statistics were gathered for each simulation.

4.1 Cache Configurations

A variety of cache configurations are used for the target machines. All L1 caches are equally split and direct mapped; all L2 caches are unified and use LRU replacement policy. All other parameters are SimpleScalar 3.0c defaults.

A direct mapped L1 and a common block size for both L1 and L2 simplifies the maintenance of an exclusive cache hierarchy. This allows for a performance comparison between equivalent exclusive and inclusive cache hierarchies. Under these circumstances, L1 hit rate is not used as a metric because it is identical across configurations.

To evaluate the exclusive cache performance, simulations using 10 sets of cache configurations were performed. For each configuration, 10 benchmarks were examined. Among these configurations, one was selected as the standard machine, whose configuration is given in Table 2.

Table 2: Standard Machine Configuration

Parameter	Configuration
L1 I-cache	32kB direct; 32B line, 1 cycle lat
L1 D-cache	32kB direct; 32B line, 1 cycle lat
L2 Unified Cache	256kB 4-way, 6 cycle lat
VB or VC entries	8 entries
Branch Prediction	2k bimodal
BTB	512 entry 4-way set assoc.
Return Addr. Stack	8 entry queue
ITLB	16 entry 4-way, 4kB mapping
DTLB	32 entry 4-way, 4kB mapping
Fetch/Issue/Commit Width	4 entries
Register Update Unit size	32 entry queue
Load / Store Queue size	8 entry queue
Mem ports (to CPU)	2 ports
Integer ALU	4 units
Integer Mult / Div	1 unit
FP ALU	4 units
FP Mult / Div	1 unit

The other configurations differ with the standard configuration in L1 cache size, L2 cache size, and VB entries respectively. Correspondingly, three groups of the configurations are defined, which are tabulated in Table 3.

Table 3: Cache configurations

Group	Config #	Characteristics
A (L2 cache size)	1	128k
	Standard	256k
	2	512k
B (L1, L2 size)	3	32k split L1, 128k L2
	Standard	64k split L1, 256k L2
	4	128k split L1, 512k L2
	5	256k split L1, 1M L2
C (VB size)	6	2 entries
	7	4 entries
	Standard	8 entries
	8	16 entries

Simulations were performed for all the benchmarks listed in Table 1 with each of the cache configurations listed in Table 3. This means simulations were performed for 10 benchmarks, 9 cache configurations each, resulting in a total of 180 simulations.

4.2 L2 Cache Size

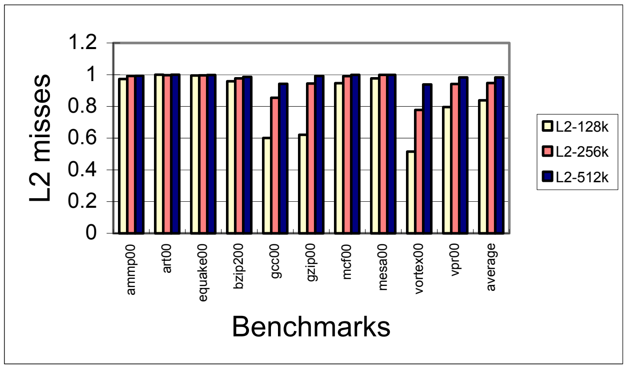


Figure 4: L2 misses comparison with varying L2 size

Figure 4 shows the L2 misses of an exclusive cache normalized to the L2 misses of an inclusive cache with varying L2 cache size from 128kB to 512kB. When the L2 cache size is relatively small (128kB), exclusive caching shows significant improvement over inclusive caching, where the average improvement is about 16%. With a larger L2 cache size, the performance improvement becomes less significant. This is because when L2 size increases, the increase in effective cache size is the same for both inclusive and exclusive caching, i.e., the ratio $\frac{effective\_cachesize_{exclusive}}{effective\_cachesize_{inclusive}}$ decreases. The minimal reduction in average L2 misses is still 1.6% when the L2 cache size is 512kB.

Increasing L2 cache size results in a less significant difference between exclusive and inclusive caches for benchmarks of *art00 and equake00*, as these benchmarks have an extremely large data set, have inherently poor cache performance, and are insensitive to cache size.

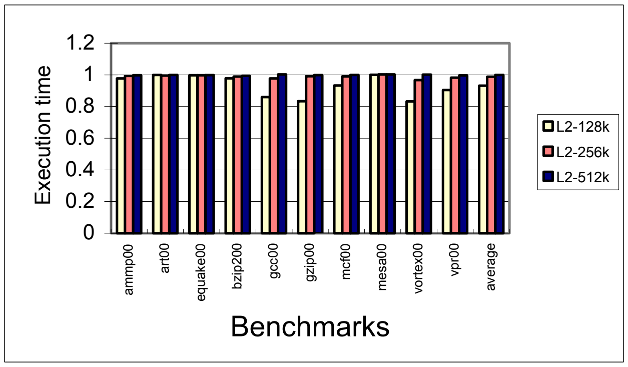


Figure 5: Execution time with varying L2 size

Figure 5 shows the execution time of benchmarks with an exclusive cache normalized to that of an inclusive cache when L2 cache capacity is increased. In general, with larger L2 cache size, the execution time for both inclusive and exclusive caching is reduced. Figure 5 illustrates that the advantage of exclusive caching is significant for many benchmarks, especially when the L2 cache size is small. The best performance gained is 16% by the gzip*00* benchmark when the L2 size is 128kB. Some benchmarks do not show significant reduction in execution time although they exhibit considerable reduction in L2 misses. In the case of *mesa00* the execution time is slightly higher when exclusive caching is used due to stalls required when the victim buffer is full.

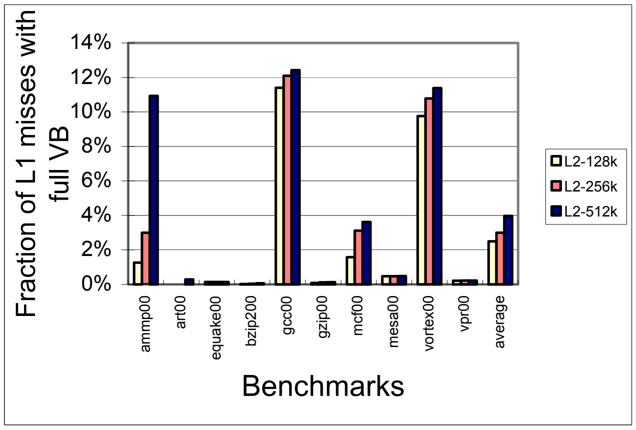


Figure 6: Rate of L1 Miss with Full Victim Buffer

Exclusive caching may reduce performance for two reasons. First, the exclusive cache has a higher worst-case penalty than the inclusive cache in the scenario where the victim buffer is full and an L1 miss occurs. From Figure 6, it can be seen that this scenario happens frequently for *ammp00*, *gcc00*, *vortex00* and *mcf00*. As L2 size increases, the worst-case latency is incurred more frequently, due to increased L2 hit rate. The high latency associated with a VB ejection is more significant than the increased capacity available through exclusive caching, yielding reduced performance for exclusive caching in some benchmarks.

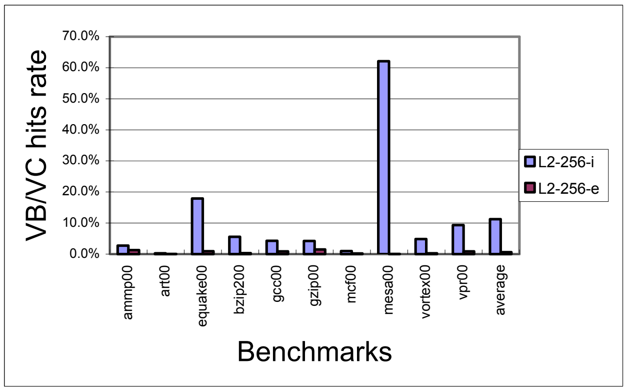


Figure 7: VB or VC hits rate

The second reason why inclusive cache execution time may be less than exclusive cache execution time is due to the hit rate of the victim cache exceeding that of the victim buffer. All inclusive caches in this study incorporate victim caches. As can be seen in Figure 7, the hit rate of the VC is much higher than the VB. For example, *mesa00* achieves a 60% hit rate for VC while only a 0.06% hit rate for VB. This gives an inclusive cache a significant benefit, especially when L1 cache size is small. Even with this inclusive cache advantage due to the victim cache, the exclusive cache organization still results in a lower execution time for most benchmarks, as can be observed in Figure 5.

4.3 L1 and L2 Cache Size

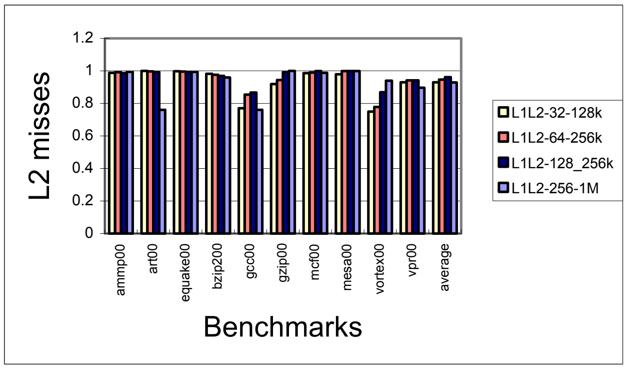


Figure 8: L2 misses with varying L1 and L2 size

Figure 8 shows exclusive cache L2 misses normalized to inclusive cache L2 misses when the L1 and L2 cache size are similarly increased. With a small L1 and L2, some benchmarks show significant performance improvement with exclusive caching. For example, L2 misses are reduced by 25% for *vortex00* and *gcc00* with small cache sizes. When L1 and L2 are sufficiently large, the L2 cache can satisfy most requests, so L2 misses will drop accordingly. From the average L2 misses shown in Figure 8 it can be seen that the performance advantage of an exclusive cache is reduced as L2 cache size increases. This is due to the reduced capacity advantage of an exclusive cache relative to an inclusive cache as the L2 cache size is increased. For a few benchmarks such as *art00* and *gcc00*, when the L1-L2 size is increased to 256k-1M, exclusive caching demonstrates a significant reduction in L2 misses. This is because these benchmarks have a large data set, and exclusive caching provides greater capacity. The average L2 misses are reduced by approximately 5% due to exclusive caching.

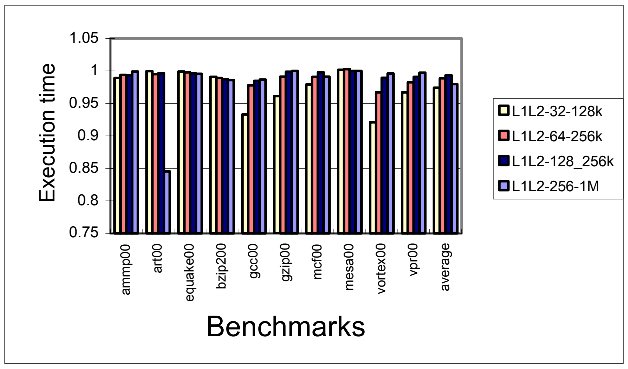


Figure 9: Execution time with varying L1 and L2 size

Figure 9 illustrates the comparison of execution time, where the execution time with an exclusive cache is normalized to the execution time with an inclusive cache. It can be seen that the pattern is similar to that of the L2 misses as shown in Figure 8. Benefits of exclusive caching diminish with increased L1 and L2 size for some benchmarks such as *gcc00* and *vpr00*. The main reason for this phenomenon is that the working set of these benchmarks is relatively small. This will cause the L2 accesses to be reduced with a large L1, making the exclusive caching less useful. Some benchmarks, such as *equake00* and *gzip00,* have better performance with a larger cache size. One reason is that the impact of a VB on execution time diminishes when cache size increases. Another reason is that a large cache size can satisfy the requirements of a large data set, and exclusive caching can provide larger effective cache size. *Art00* realizes a performance improvement of 15.5% with an exclusive cache hierarchy when L1 and L2 size are 256kB and 1MB respectively. This is due to the significant reduction of L2 misses. The average execution time reduced by exclusive caching is about 2.5% with a small cache size and 2% with a large cache size.

4.4 Victim Buffer Size

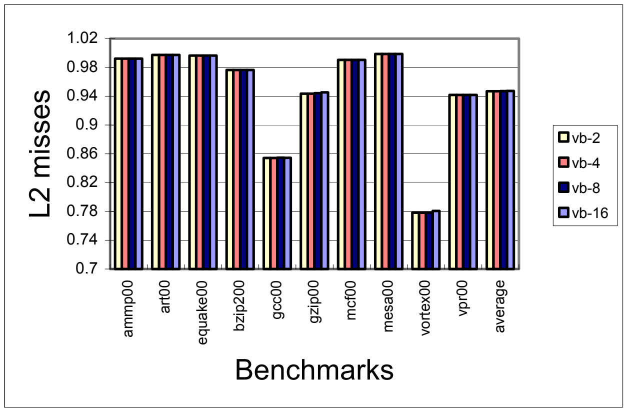


Figure 10: L2 misses with varied VB entries

Figure 10 shows the L2 misses of an exclusive cache normalized to the L2 misses of an inclusive cache with VB entries varied from 2 to 16 entries. Each VB entry has the same size as a cache line. For exclusive cache, the victim buffers may theoretically service some L1 misses, resulting in fewer L2 accesses. However, this caching function of a VB is secondary to the enabling of processor accesses to be serviced without waiting for an L1 victim to be flushed back to the L2 cache. The hit rate of the victim buffer varies from 0% to 1.6%, whereas the hit rate of a victim cache varies from 0% to 60% as is seen in Figure 7. The victim cache reduces misses more significantly than victim buffer does, but most of those misses are also available in the L2 cache. Thus it can be seen that there is little difference in L2 misses, as shown in Figure 10, when VB entries are varied from 2 to 16.

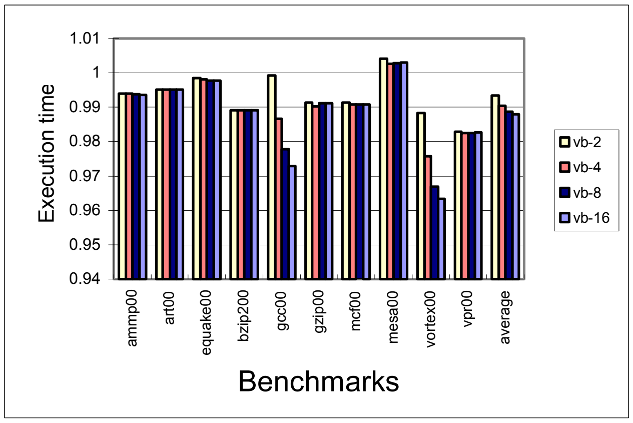


Figure 11: Execution time with varied VB entries

Figure 11 shows the execution time of an exclusive cache normalized to that of an inclusive cache with increasing victim buffer entries. If the victim buffer has more than four entries, there is sufficient idle bus time to drain the victim block from the victim buffer, and the worst-case scenario does not introduce significant penalties to the exclusive cache execution time. With smaller L1 cache, the number of VB entries needs to be increased in order to satisfy an increased number of accesses. Correspondingly, as L1 cache size increases, the number of VB entries can be smaller since the accesses would be reduced, and fewer entries are sufficient to meet the requirement.

5 CONCLUSIONS

This research is motivated by the increasing performance differential between processor and main memory (i.e. DRAM). Exclusive caching, like all caching, serves to reduce the dependence of the processor upon data residing in the main memory. This furthermore reduces average access latency and bus traffic on limited bandwidth interconnects such as the front-side and DRAM bus. Exclusive caching has many advantages over conventional inclusive caching [5]. Increased associativity is provided through the increased capacity as two memory references that are mapped to the same cache set can reside in either the L1 cache or the L2 cache. Increased hit rate is also made possible by the increased cache capacity which is better utilized, since there are no duplications between the contents of the L1 cache and L2 cache.

This paper illustrates the effects of an exclusive two-level cache memory hierarchy with SPEC 2000 benchmarks. The cache configurations varied in L2 size, L1 and L2 size, and the number of victim buffer entries. The results of the simulations are encouraging: exclusive caching provides benefits to most benchmarks, especially when L2 cache size is small. As the L1 and L2 cache sizes increase, the advantages of exclusive caching become less noteworthy, especially for benchmarks with a large working set. For systems that have small cache sizes, running applications with good caching behavior (high cache hit rate), the performance improvement attributable to exclusive caching is most significant.

Some applications such as *mesa00*, which have a large amount of L1 conflict misses, demonstrate better performance with an inclusive cache hierarchy. There are two reasons. First, the victim cache in an inclusive hierarchy has a very high hit rate for these benchmarks. Second, the worst-case latency in an exclusive cache hierarchy occurs more frequently when the victim buffer has small size. These observations explain why the execution time of an inclusive cache hierarchy may be less than that of an exclusive cache hierarchy.

The results of our simulations yield that the number of victim buffer entries has little impact upon performance beyond 4 victim buffer entries. This assumes a relatively large 64kB split L1 cache. The victim buffer can act as a victim cache and reduce the L2 cache accesses. However, in the simulations performed, the hit rate of the victim buffer never exceeded 1.6% regardless of the number of victim buffer entries. If the L1 cache size is increased, the victim buffer entries involved can be correspondingly smaller to supply both higher utilization and equivalent performance.

One limitation of the two-level exclusive cache hierarchies simulated herein is that the cache block size is constrained to be identical for both the L1 and L2 cache, making the design space less flexible. It is common to make the L2 cache line larger than the L1 cache line, and if all caches (L1, L2, VB) use the same line size, each cache design will be less flexible and this may result in reduced performance. For this reason it is suggested that a prefetch buffer is used in combination with an exclusive cache utilizing a small or medium block size. A second drawback of exclusive caching is that increased data movement among non-redundant caches requires additional control and incurs additional power consumption. A third drawback is that to implement exclusive caching in SMP requires more chip area to implement L1 snoop ports, which increases the cost. None of these criticisms of exclusive caching has been directly addressed in these simulations.

Considering the complexity involved in an exclusive cache hierarchy and the current silicon technology, the exclusive cache hierarchy is suitable for server applications that perform a large amount of memory accesses and embedded systems that have limited silicon space for cache and memory. This study is limited in that it only examines the single-threaded application workload. Future work includes applying exclusive caching in multithreaded applications and multi-processor systems.

REFERENCES

[1] A.J.Smith, "Cache Memories," ACM Computing Surveys, Vol.14, No.3, September 1982.

[2] J.L.Baer, and W.H.Wang. "On the inclusion properties for multi-level cache hierarchies," *Proceedings of the 15th Annual International Symposium on Computer Architecture*, page 73-80, 1988.

[3] ADVANCED MICRO DEVICES, INC. "AMD Athlon^{TM} Processor and AMD Duron^{TM} Processor with full-speed on-die L2 cache," June 19, 2000.

[4] T.M.Wong, Gregory R.G, J.Wilkes, "My cache or yours? Making storage more exclusive," November 2000.

[5] N.P.Jouppi and Steven J.E.Wilton, "Tradeoffs in Two-Level On-chip Caching," WRL Research Report 93/3, October 1993.

[6] "SimpleScalar LLC", http://www.simplescalar.com (Current June 2003).

[7] SPEC, "SPEC CPU2000 Benchmarks," Standard Performance, Evaluation Corporation, 2000. http://www.spec.org/osg/cpu2000/ (Current June 2003).

[8] D.E.Culler, J.P.Singh, A.Gupta. "Parallel Computer Architecture, A Hardware Software Approach," 1998, p396.

[9] J.L.Hennessy and D.A.Patterson, "Computer Architecture: A Quantitative Approach," Morgan Kaufmann Publishers, 2nd edition, 1996.

[10] B.T.Davis, *Modern DRAM Architectures,* doctoral dissertation. Dept. Electrical Engineering and Computer Science, Univ. of Michigan, Ann Arbor, Nov. 2000.

[11] J.J.Johnson, " The AMD-760^{TM} MPX Platform for the AMD Athlon^{TM} MP Processor," January 4, 2002.

[12] N.P.Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," *Proceedings of the 17th Annual International Symposium on Computer Architecture*, p.364-373, May 1990.

[13] Intel Corporation, http://www.intel.com (Current June 2003).

[14] Advanced Micro Devices, AMD, http://www.amd.com/us-en (Current June 2003).

[15] Ying Zheng, *Exclusive Cache Architecture and Performance Evaluation*, MS thesis, Dept. Electrical and Computer Engineering, Michigan Tech, Houghton, May. 2003.