This article has been accepted for publication in Computer but has not yet been fully edited.
Some content may change prior to final publication.

Concurrent Average Memory Access Time

Xian-He Sun Dawei Wang
Illinois Institute of Technology
sun@iit.edu, david.albert.wang@gmail.com

ABSTRACT
Concurrency is a common technique used in modern memory systems. However, the effectiveness of memory concurrency is application dependent. It varies largely from application to application and from implementation to implementation. Understanding and utilizing memory concurrency is a vital and timely task for data intensive applications. Traditional memory performance metrics, such as Average Memory Access Time (AMAT), are designed for sequential data accesses, and have inherent limitations in characterizing concurrency. In this study, we propose Concurrent Average Memory Access Time (C-AMAT) as an accurate metric for modern memory systems. C-AMAT has the ability to examine the impact of concurrent memory behavior at both the component and system level of modern memory systems. It is a good guide for design choices, while other conventional memory metrics often mislead in measurement when concurrency is present.

Keywords
Memory concurrency; memory metric; memory performance measurement

1. INTRODUCTION
The unbalanced technological advancements in processor and memory over the past thirty years have led directly to the "memory wall" problem [1] [2], which means there is an ever increasing gap between CPU and memory performance. Cache hierarchies are the most effective mechanism for hiding the long delay of off-chip main memory accesses. As the "memory wall" problem becomes worse, the long delays exist not only in main memory, but also penetrate into the cache hierarchies. For instance, in the Intel Nehalem architecture CPU, each L1 data cache has a 4-cycle hit latency; and each L2 cache has a 10-cycle hit latency. Additionally, the IBM Power6 has a 4-cycle L1 cache hit latency, and a L2 cache hit latency of 24 cycles. For Last Level Caches (LLC) of modern multi-core processors, latencies can even exceed one hundred cycles. In order to alleviate the effect of these performance gaps, intensive research has been conducted to improve the concurrency of memory systems. Multi-port cache, multi-banked cache and pipelined cache [3] are advanced cache design techniques which improve cache hit concurrency; whereas, non-blocking cache [4] is the main technique to improve cache miss concurrency. Processor ILP (Instruction Level Parallelism) techniques, such as out-of-order execution, multiple issue pipeline, SMT (Simultaneous Multi-Threading), CMP (Chip Multi-Processing), etc., can dramatically improve both cache hit and miss concurrency. With these advanced cache optimizations and processor ILP techniques, tens of or even hundreds of cache/memory accesses may coexist in the memory hierarchy at the same time. Thus, a single cache miss is no longer a determine factor of the overall memory system performance. However, existing memory metrics, such as Miss Rate (MR), Average Miss Penalty (AMP), and Average Memory Access Time (AMAT), are still measured based on single access activity, which does not reflect the reality of cache/memory concurrency. While it becomes increasingly important, understanding the contributions of various concurrencies of modern memory systems remains elusive for researchers and practitioners. In this study, we extend concurrency to the widely-used AMAT metric to create the Concurrent AMAT (named C-AMAT in short) metric. Similar to AMAT, C-AMAT formulates the memory access delay into a summation of memory hierarchy access delays. It introduces two new average concurrency parameters at each level of the memory hierarchy, namely hit concurrency and miss concurrency. C-AMAT argues that hit concurrency will improve performance while a cache miss may or may not reduce the overall memory system performance, depending on hit concurrency. When the sum of the two concurrencies equals to one, that is when the data access is sequential, C-AMAT is the same as AMAT. Similar to AMAT, C-AMAT is a powerful tool to evaluate architecture design choices, from hit/miss ratio to hit/miss concurrency.

APC (memory Access Per Cycle) is a new performance metric designed to measure concurrent memory system performance [5]. First, through proving the C-AMAT formulation is the reciprocal of the APC measurement [5], we confirm the correctness of the C-AMAT definition. Then, we exam the relation between C-AMAT and AMAT, and formally introduce the associated hit and miss concurrency definition. Finally, we examine the impact of the configuration of processor microarchitecture and cache using C-AMAT and other existing memory performance metrics. Experimental results confirm that only C-AMAT always captures the characteristic of modern commercial processors, whereas conventional memory metrics often provide misleading information.

2. DEFINITION AND FORMULATION

2.1 Definition
C-AMAT is defined as the average memory access time with the consideration of concurrent hit and miss accesses. Quantitatively speaking, C-AMAT is equal to the total memory access cycles divided by the total number of memory accesses. Let $T_{MemCycle}$ represent the total number of cycles executed in which there is at least one outstanding memory reference; let C_{MemAcc} represent the total number of memory accesses. Therefore

$$C - AMAT = \frac{T_{MemCycle}}{C_{MemAcc}} \quad (1)$$

The definition of C-AMAT is very simple. The tacit concurrency of Eq.(1) is that the memory cycles are parallel cycles. That is when there are several memory accesses co-existing during the same cycle, $T_{MemCycle}$ only increases by one. On other words, when counting memory access cycles, an *overlapping* mode is

Digital Object Indentifier 10.1109/MC.2013.227 0018-9162/$26.00 2013 IEEE

adopted. Due to the advanced structures of modern cache and memory systems, such as pipelined cache, multi-ported cache, non-blocking cache, etc., multiple hit accesses and miss accesses could be overlapped with each other. $T_{MemCycle}$ catches this concurrency. Another important feature of $T_{MemCycle}$ is that $T_{MemCycle}$ only includes the clock cycles with memory access activities; the cycles without memory references are excluded. According to the definition of APC (memory Access Per Cycle) [5], the C-AMAT is the reciprocal of APC. Based on the measurement methodology of APC proposed in [5], the final value of C-AMAT can be obtained.

$$C - AMAT = \frac{1}{APC} = \frac{T_{MemCycle}}{C_{MemAcc}} \qquad (2)$$

Paper [5] used the statistical variable *correlate coefficient* to verify the correctness and accuracy of APC, and introduced a practical method to measure APC. The direct relation of C-AMAT and APC translates the correctness, accuracy, and feasibility of APC to C-AMAT.

Eq. (2) provides the correctness of C-AMAT. Correctness, however, is only part of the story. C-AMAT is an analytic tool. Like AMAT, a component-based, parameterized formula of C-AMAT needs to be derived in order to put C-AMAT in use.

2.2 Extend AMAT Formula with Concurrency

The traditional AMAT is calculated as HitCycle+MR×AMP. HitCycle (marked as *H* in equation 3) is the hit time of memory accesses; MR is the miss rate of cache accesses; and AMP is the average miss penalty. AMP is calculated as the sum of all single miss access latency divided by the total number of miss accesses. AMAT does not consider the concurrency of memory accesses, in either the hit or the miss section of the formula. It assumes that the memory accesses are sequential, one after another. Furthermore, with concurrent accesses, hits and misses may co-exist at the same cycle. The sequential view of AMAT is simple and worked well in the past, but is impeded toward modern processor architectures and memory systems where concurrency is paramount. To properly analyze the concurrency, we extend AMAT with concurrency parameters for hit and miss accesses, and propose a new counting method for MR and AMP which considers the relation between concurrent hits and misses. The extended formula is shown in (3).

$$\frac{H}{C_H} + MR \times \frac{AMP}{C_M} \qquad (3)$$

The first parameter C_H represents the hit concurrency; the second parameter C_M represents the miss concurrency. The C_H could be contributed by multi-port cache, multi-banked cache or pipelined cache structures. The C_M could be contributed by non-blocking cache structure. In addition, processor ILP design techniques, such as out-of-order execution, multiple issue pipeline, SMT, CMP, etc., can both increase the hit concurrency and miss concurrency. The Miss Rate in formula (3) is re-defined as the number of pure misses over the total number of accesses. The pure miss here means that the miss contains at least one miss cycle which does not have any hit access activity. When measuring private caches for CMP processors, e.g. L1 data cache, the pure misses are measured based on "per-core" mode, which means every core has its own detecting logic, and that logic only measures that core's private cache accesses. When a miss occurs without a hit access inside the private cache, the correspondent cycle is measured as a "pure miss cycle" for that core. For shared caches, e.g. L2 or L3 caches, the pure miss cycles are measured based on "all-core" mode, which means when there is no cache hit access from any of the cores, then a miss cycle is counted as "pure miss cycle". AMP is also re-defined as the average number of pure miss cycles per miss access.

C-AMAT can be calculated using hit and miss concurrency factors for architecture design choices. The critical question is how to obtain an accurate average C_H and C_M. Here a weighted method is applied to calculate the average value.

Let C_H be the average hit cache concurrency, by definition it is equal to

$$C_H = \sum_{i=0}^{N} C_i \times \frac{t_i}{T_H}$$

N is the total number of cache hit phases. In each hit phase the value of C_i does not change; C_i is the hit concurrency during phase *i*; t_i is the number of cycles of phase *i*. Note these hit access phases only include cache cycles containing at least one cache hit activity, clock cycles without any hit accesses cannot be counted into a hit access phase. T_H is the total hit cycles in the overlapping mode, therefore

$$T_H = \sum_{i=0}^{N} t_i$$

Similar, the definition of C_M can be given as below,

$$C_M = \sum_{j=0}^{M} C_j \times \frac{t_j}{T_M}$$

M is the total number of pure cache miss phases. In each miss phase, the value of C_j does not change; C_j is the miss concurrency during the phase *j*; t_j is the number of cycles of phase *j*. Note that the pure miss phases only include the cache cycles that contain at least one pure cache miss activity. If one clock cycle contains miss access as well as hit access or does not contain any miss access, this cycle is not counted in pure miss phases. T_M is the total pure miss cycles in the overlapping mode, therefore

$$T_M = \sum_{j=0}^{M} t_j$$

In the following, we will analytically prove formula (1) is equal to formula (3) based on their definitions and show how the C-AMAT formulation (3) can be extended recursively from L1 cache down to LLC.

2.3 Proof of Equality

In order to prove formula (1) and formula (3) are equivalent, more detailed descriptions of the related variables are listed below.

H is the number of hit cycles when accessing the current cache layer. Every cache access needs to spend *H* cycles to determine whether this is a hit or a miss access. Note *H* is a constant value in our cache model.

Digital Object Indentifier 10.1109/MC.2013.227 0018-9162/$26.00 2013 IEEE

MR (Miss Rate) in this study is an extended version of the traditional miss rate definition with the consideration of concurrency. Only when a miss access has no overlapping with any hit accesses, then this miss access is a pure miss access. Thus,

$$MR = \frac{C_{MemPMiss}}{C_{MemAcc}}$$

$C_{MemPMiss}$ is the total number of pure misses.

AMP is the average miss penalty which only considers pure miss accesses.

$$AMP = \frac{T_{MemPMiss}}{C_{MemPMiss}}$$

$T_{MemPMiss}$ is the sum of total pure miss cycles. The pure miss cycles are the cache miss access cycles without any hit access. Thus

$$\frac{H}{C_H} + MR \times \frac{AMP}{C_M} = \frac{H}{\sum_{i=0}^{N} C_i \times \frac{t_i}{T_H}} + \frac{C_{MemPMiss}}{C_{MemAcc}} \times \frac{T_{MemPMiss}}{C_{MemPMiss}} \times \frac{1}{\sum_{j=0}^{M} C_j \times \frac{t_j}{T_M}}$$

$$= \frac{H \times T_H}{\sum_{i=0}^{N} C_i \times t_i} + \frac{T_{MemPMiss}}{C_{MemAcc}} \times \frac{T_M}{\sum_{j=0}^{M} C_j \times t_j} \quad (4)$$

Because,

$$\sum_{i=0}^{N} C_i \times t_i = C_{MemAcc} \times H$$

$$\sum_{j=0}^{M} C_j \times t_j = T_{MemPMiss}$$

Thus,

$$(4) = \frac{H \times T_H}{C_{MemAcc} \times H} + \frac{T_{MemPMiss}}{C_{MemAcc}} \times \frac{T_M}{T_{MemPMiss}}$$

$$= \frac{T_H + T_M}{C_{MemAcc}} = \frac{T_{MemCycle}}{C_{MemAcc}} = C - AMAT \quad \blacksquare$$

So,

$$C - AMAT = \frac{H}{C_H} + MR \times \frac{AMP}{C_M} \quad (5)$$

According to formula (5), there are five important parameters determining the overall memory performance, namely hit latency, hit concurrency, miss rate, average miss penalty, and miss concurrency. Though the concepts of miss rate and average miss penalty in C-AMAT are similar to the counterparts used in AMAT, C-AMAT excludes hit and miss accesses overlapping cycles.

Without considering concurrency, the traditional memory metrics often provide an unhelpful measurement, which may lead to improper design considerations. Only by considering concurrency will the memory performance reflect the impact of micro-architectural design choices correctly. Also as we will see in section 5, different processor and cache design choices will affect different C-AMAT parameters, sometimes a technique might improve one factor while deteriorating others. These parameters in turn affect the overall data access time, and therefore determine the design choices. Only comprehensively considering all factors can determine the most appropriate system configuration of a memory system.

3. CACHE CONCURRENCY MEASUREMENT

As introduced in formula (5), two concurrencies exist at each layer of a memory hierarchy, namely hit concurrency and miss concurrency. Hit concurrency reflects the parallelism of cache tags query and cache data access. It does not matter whether a cache access finally turns out to be a hit or a miss access; this cache access requires the system to spend a certain fixed cycles performing a cache tags query. Considering advanced cache design techniques such as multi-ported cache and pipelined cache, the maximum hit cache concurrency is (#cache port × #cache pipeline stage). For example, the AMD Opteron CPU has a two-port L1 data cache, and a 3-cycle pipeline stage for cache access [6], thus the maximum hit concurrency is 2×3=6. The miss concurrency is usually determined by the number of MSHR (Miss Status Holding Register) entries. The maximum miss concurrency is equal to the number of outstanding cache misses that MSHR can support.

Compared with the traditional AMAT miss rate measurement, C-AMAT does not include any miss cycles which overlap with a hit cycle. Because when a hit occurs, the memory does not block CPU performance. Only pure miss access cycles cause the CPU to discontinue execution. Thus, the challenge of measuring miss concurrency is the elimination of overlapping cycles which contain hit accesses. In other words, the miss concurrency detector needs to simultaneously be aware of both cache hit accesses and miss accesses. Only pure miss cycles, defined as miss cycles which do not overlap with hit cycles, are counted in C-AMAT as miss cycles. A detailed cache hit and miss concurrency detecting structure is shown in Fig. 1.

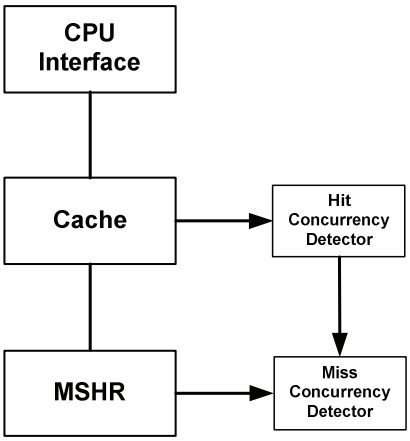

Figure 1. C-AMAT Detecting Structure

The Hit Concurrency Detector (HCD) counts the total hit cycles and records each hit phase in order to calculate the average hit concurrency. The hit cycles are the clock cycles containing at least one hit access activity. The HCD also tells the Miss Concurrency Detector (MCD) whether a current cycle has a hit access or not. The MCD is a monitor unit which counts the total number of pure miss cycles and records each pure miss phase in order to calculate the average miss concurrency, pure miss rate, and pure miss penalty. With the information provided by the HCD, the MCD is able to tell whether a cycle is a pure miss cycle, and whether a miss is pure miss. Furthermore with all miss information, the pure miss rate and average pure miss penalty can be calculated. Finally with formula (5), C-AMAT can be measured with the five parameters at the right-side of the equation. Similar to AMAT, the AMP in Eq. (5) can be further extended as a composition of the hit and miss of the next level cache of the memory hierarchy. The extension is straightforward.

4. MICROARCHITECTURE IMPACT

To demonstrate their contributions to cache concurrency, several processor microarchitectures and advanced cache design choices are presented herein. From a processor microarchitecture aspect, critical ILP technologies, such as out-of-order execution, multiple issue pipeline, load/store unit, issue buffer and reorder buffer, have not only a large impact on instruction parallelism, but also on cache/memory access concurrency. SMT allows a processor to execute multiple threads' instructions in one cycle; while CMP allows the execution of multiple independent applications concurrently within different processor cores on the same die. Both of these techniques can dramatically increase cache hit and miss concurrency. Advanced cache design techniques, such as multi-port cache, multi-banked cache, pipelined cache, can directly affect maximum cache hit concurrency; other cache advanced technologies, such as non-blocking cache, cache prefetching, main memory bank-level parallelism, greatly influence cache miss concurrency.

We select three basic and fundamental techniques to verify the accuracy and practical usefulness of C-AMAT. They are multiple issue pipeline, non-blocking cache, and chip multi-processing.

Multiple Issue Pipeline

Multiple issue pipeline allows multiple instructions to be fetched, decoded, issued, executed and committed in the same cycle. It is a very important ILP design technique widely used in modern commercial processors. Due to data dependency of algorithms, high penalty of branch mis-prediction, missing load data, and "power wall" problems, over-increasing pipeline width is not a good design choice. Usually commercial processors adopt a 4 to 8 width pipeline at different stages [3]. In this study, we assume each stage has the same width for simplicity.

Non-blocking Cache

To cooperate with modern processor technologies, such as out-of-order speculation, multiple issue, multi-threading, and multi-core technologies, modern CPUs, such as Intel Core, Itanium [7], and IBM POWER3, employ non-blocking cache heavily at each level of a memory hierarchy in order to enhance memory access parallelism. Non-blocking caches can continue supplying data under certain number of cache misses by adopting a Miss Status Holding Register (MSHR) [4]. The MSHR is a structured table, which records cache miss information such as access type (load/store), access address, and return register, etc. When the MSHR table is empty, there are no outstanding cache misses. When the MSHR is attached to LLC (Last Level Cache) and empty, there are no outstanding main memory accesses. When the MSHR table is full, the cache cannot afford more cache accesses, and the CPU's memory accesses or next-level memory accesses are blocked. Therefore, the number of MSHR entries can directly determine miss access concurrency.

Chip Multi-Processor

Due to the "power wall" problem and limited performance gain from ILP technologies, increasing the number of processor cores on the same die is becoming the most important and efficient method of increasing total system performance. In CMP processor, different cores share a Last Level Cache. Increasing the number of cores can directly increase LLC hit and miss concurrency.

5. EXPERIMENT TESTING

A detailed CPU model in the GEM5 simulator [8] was adopted, which supports out-of-order, speculative execution, superscalar, and multi-threading for a single core, and complicated cache hierarchies for multi-cores with different cache coherency protocol. Unless stated otherwise, the experiments assume the following default processor and cache configuration showing in Table I.

Table 1. Default Simulation Configuration Parameters

Parameter	Value
Processor	1core, 4 GHz, 4-issue width,
Function units	6 IntALU 1 cycle, 1 IntMul 3 cycles, 2 FPAdd 2 cycles, 1 FPCmp 2 cycles, 1 FPCvt 2 cycles, 1 FPMul 4 cycles, 1 FPDiv 12 cycles
ROB, LSQ size	ROB 64, LQ 48, SQ 24
L1 caches	32KB Inst/32KB Data, 2-way, 64B line, hit latency: 4 cycle Inst/4 cycle Data, ICache 8 MSHR Entry, DCache 8 MSHR Entry
L2 cache	512KB, 16-way, 64B line, 24-cycle hit latency, 16 MSHR Entry
DRAM latency/Width	240-cycle access latency/64 bits

In this study, there are three different sets of configurations based on the default configuration. Each of them only changes one or two parameter of the simulation in order to show the influence of each design choice on the cache concurrency and C-AMAT. The three design choices, as stated in Section 4, are multiple issue pipeline, non-blocking cache, and multi-core design choices. The variation of all memory performance metrics, including MR, AMP, AMAT and C-AMAT were used to distinguish the correctness of each memory performance metric. The winner should always correlate to processor design choices.

The simulations were conducted using 24 benchmarks from SPEC CPU2006 suite [9]. Some benchmarks in the set were omitted because of compatibility issues with the simulator. The benchmarks were compiled using GCC 4.3.2 with -O2 optimization. The reference input sizes provided by the benchmark suite were adopted for all benchmarks. For each benchmark, 10 million instructions were simulated to collect statistics. The memory performance results presented are the average value for all 24 benchmarks. The advantage and accuracy

Digital Object Indentifier 10.1109/MC.2013.227 0018-9162/$26.00 2013 IEEE

of the C-AMAT measurement already verified in [5] in terms of APC. The experimental testing herein is focused on the credential and usefulness of C-AMAT formulation in evaluating memory design choices. The measurement and discussion are for a general purpose memory system design. For a special designed memory system for a particular benchmark (application), the measurement and discussion can be narrowed in for that particular benchmark.

By comparing results of C-AMAT and AMAT for each design choice, it is clear that only C-AMAT always matches actual design choices for modern processors.

5.1 Impact of Multiple Issue Pipeline Width

Multiple issue pipeline is an important processor microarchitecture design to improve ILP. Most modern general purpose CPUs have an issue width between 4 and 8. From Figure 2 (b)~(d), it can be observed that only the memory performance reflected by C-AMAT can correctly match the performance improvement trend of practical processor design considerations. Other memory performance metrics, including Miss Rate, Average Miss Penalty, and AMAT do not correctly reflect the memory performance variation trend.

Fig. 2(a) shows the average hit concurrency, and average pure miss concurrency of an L1 data cache as issue width is increased from 1 to 8. It can be observed that the average hit concurrency increases at a greater rate than average pure miss concurrency when the pipeline width is larger than 4. According to Fig. 2(b), the traditional AMAT increases when the pipeline width increases, which means the memory performance is decreasing. This is a contradiction with the fact that application performance should be improved with the increased width of issue pipeline width [3]. On the contrary, C-AMAT not only correctly describes that the overall memory performance is increasing, but also correctly reflects the diminishing returns which occur when the issue width of the pipeline is larger than 4. It is interesting to note that this C-AMAT result confirms that the current design methodology of choosing an issue width size of between 4 and 6 is optimal for general purpose processors. This demonstrates the practical usefulness of C-AMAT. According to Fig. 2(c) and Fig. 2(d), the MR and AMP are also ineffective at measuring memory performance. Only the comprehensive memory metric C-AMAT which considers hit and miss access delay, proportion and concurrency can correctly reflect the overall memory performance.

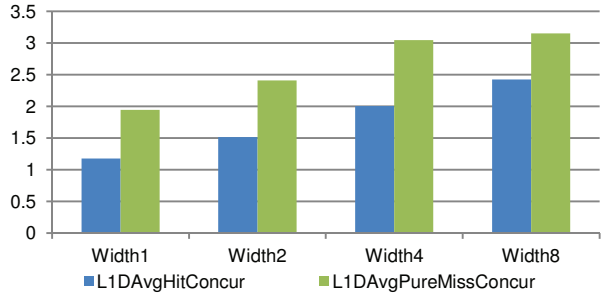

Figure 2(a). L1 DCache Concurrency when Changing Issue Pipeline Width

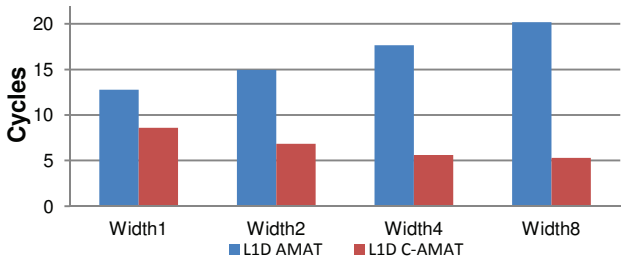

Figure 2(b). L1 DCache AMAT and C-AMAT when Changing Issue Pipeline Width

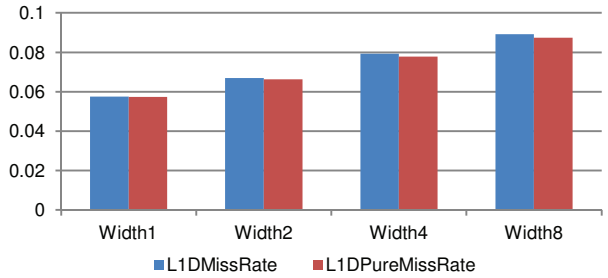

Figure 2(c). L1 DCache Miss Rate and Pure Miss Rate when Changing Issue Pipeline Width

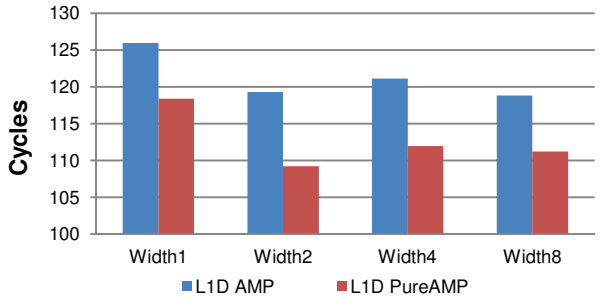

Figure 2(d). L1 DCache AMP and Pure AMP when Changing Issue Pipeline Width

Please notice that the difference between MR and Pure MR, AMP and Pure AMP are very small. For miss rate the average difference is 1.2%, for miss penalty the difference is 7.1%. The correctness of C-AMAT is mainly determined by the concurrency of memory accesses. That is reasonable, since changing pipeline issue width changes concurrency.

In the following sub-sections, we demonstrate that different architecture choices have different impacts on the five parameters of the C-AMAT equation. All the five parameters, therefore, are essential for modeling the overall memory performance. C-AMAT adds some complexity to AMAT, but the complexity is a necessity for understanding and analyzing the overall performance of modern memory systems.

5.2 Impact of MSHR Size

The size of MSHR table can directly determine the maximum miss concurrency; however, it does not have direct impact on hit concurrency. As shown in Fig. 3(a), when the number of MSHR entry is increased, the average hit concurrency stays approximately the same, whereas the average pure miss concurrency constantly increases. According to Fig. 3(b), the larger the MSHR table is, the smaller the miss concurrency gain; similarly when the number of MSHR entry increases, C-AMAT decreases while AMAT increases. This AMAT increase inappropriately describes a memory performance decrease; but C-

Digital Object Indentifier 10.1109/MC.2013.227 0018-9162/$26.00 2013 IEEE

AMAT accurately portrays the performance improvement create from the wide adoption of the non-blocking cache technique in modern processor design [7]. In the meantime, AMAT fails miserably in describing the concurrency variation of MSHR.

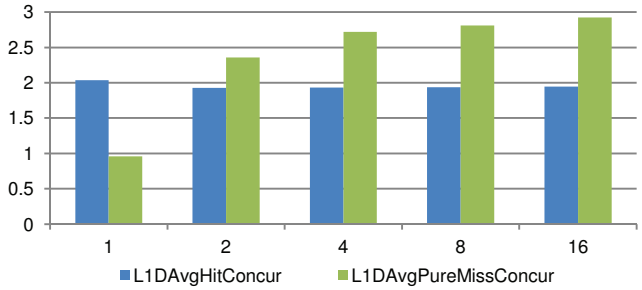

Figure 3(a) L1 DCache Concurrency when Changing MSHR Size

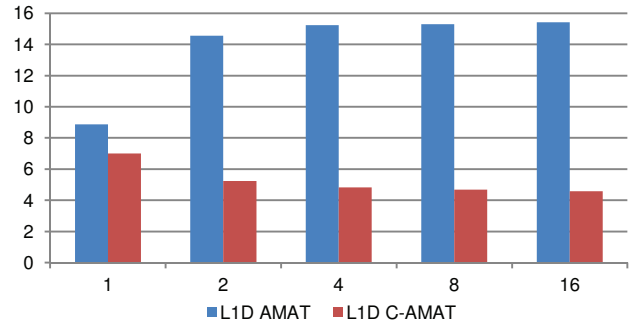

Figure 3(b) L1 DCache AMAT and C-AMAT when Changing MSHR Size

5.3 Impact of the Number of Cores

In CMP processors, several different cores on the same die share one Last Level Cache (LLC). Increasing the number of cores can directly increase LLC access concurrency. In this paper, we assume L2 is the last level cache. When changing core number with value of 1→2→4→8, L2 cache size changes according to core number with value of 512KB→1MB→2MB→4MB. According to Fig. 4(a), the L2 cache concurrencies increase with the number of cores. The miss rates are also increased due to the increase of conflicts (see Fig. 4(b)). Without considering concurrency, the AMAT metric suggests that the overall performance of the L2 cache has decreased. However, this obviously conflicts with the widespread use of multi-core processors. Under C-AMAT, the overall memory performance of L2 cache shows an increase. This observation again illustrates the importance of concurrency in data accesses.

More performance comparison of C-AMAT and AMAT can be found in [10]

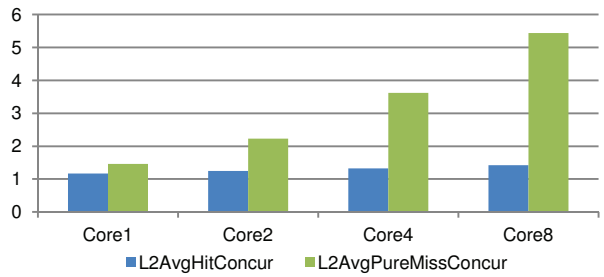

Figure 4(a) L2 Cache Concurrency when Changing Core Number

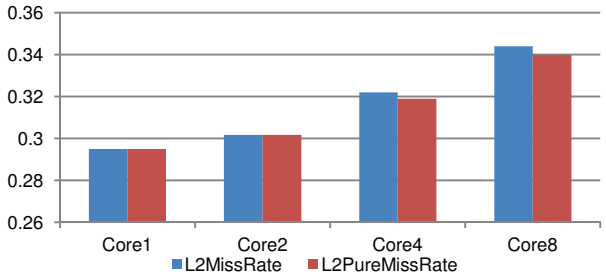

Figure 4(b) L2 Cache Miss Rate when Changing Core Number

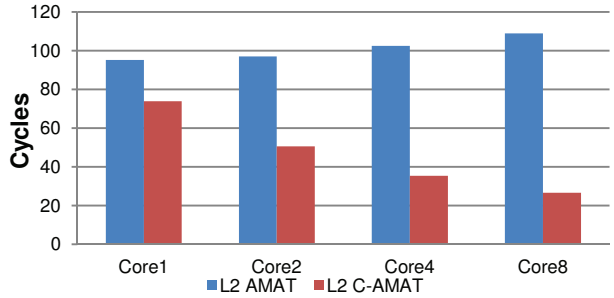

Figure 4(c) L2 Cache AMAT and C-AMAT when Changing Core Number

5.4 Related Work

There are three commonly used performance metrics for evaluating memory systems [3], namely Miss Rate (MR), Average Miss Penalty (AMP), and Average Memory Access Time (AMAT). MR only reflects the proportion of the data in or out of the cache; it does not reflect the penalty of the miss access when memory concurrency exists. AMP only catches the penalty of the cache miss access for that particular data access; it doesn't show the performance degradation of the memory system when memory concurrency exists. AMAT is a comprehensive memory metric, but it is still based on the single data access viewpoint. It does not consider the memory hit and miss access concurrency. Memory-bandwidth is usually used to measure the peak performance of memory systems with intensive memory workloads. Similar to APC [5], memory-bandwidth is a useful measurement, but lacks a mathematical formulation for analytical study. C-AMAT is an extension of AMAT to consider memory concurrency, which makes it easy to adopt for anyone already familiar with AMAT. It is a useful analysis tool to evaluate the design choices of modern memory systems.

6. CONCLUSIONS

AMAT is a widely-used conventional tool for architecture analysis and design. In this study, we argue that memory concurrency has become a vital factor of memory performance and introduce an extended AMAT, the C-AMAT (Concurrent AMAT), to consider data access concurrency in modern memory systems. We have theoretically proved the correctness of the C-AMAT formulation, and presented the measurement methodology of C-AMAT. Extensive simulations were conducted to confirm that C-AMAT is effective in evaluating modern memory system design and architecture configuration. C-AMAT evaluation leads to the same optimum design choices as the rule of thumb used in today's industry. That demonstrates the practical value of C-

Digital Object Indentifier 10.1109/MC.2013.227 0018-9162/$26.00 2013 IEEE

AMAT. C-AMAT is an extension of AMAT. When access concurrency remains unchanged, C-AMAT produces the same results as AMAT. However, when the memory access improvement comes from concurrency by using ILP or other advanced cache design techniques, such as multiple issue pipeline, non-blocking cache, etc., AMAT cannot correctly reflect the memory performance changes, and often provides misleading information, while C-AMAT can understand the improvements due to concurrency. AMAT is known for its simplicity. But, it is designed to characterize memory hierarchy, not concurrency. Concurrency has become pervasive technique in modern memory systems. This study presents a compelling investigation in the vital role of concurrency in current memory systems, and in how to evaluate the increasing complexity created by concurrency. C-AMAT has a unique importance in practical and analytical architectural study, for today and for tomorrow.

Acknowledgment

This research was supported in part by National Science Foundation under NSF grant CCF-0621435, CNS-0751200, and CCF-0937877.

7. REFERENCES

[1] X.-H. Sun, and L. Ni, Another View on Parallel Speedup, Proc. of IEEE Supercomputing'90, NY, Nov. 1990.

[2] W. Wulf and S. McKee. Hitting the wall: Implications of the obvious. ACM SIGArch Computer Architecture News, Mar. 1995.

[3] J. L. Hennessy and D. A. Patterson. Computer Architecture: A Quantitative Approach. Morgan Kaufmann, 4th edition, September 2006.

[4] David Kroft, "Lockup-free Instruction Fetch/Prefetch Cache Organization", in Proceedings of the 8th annual symposium on Computer Architecture (ISCA '81), Los Alamitos, CA, USA, 1981.

[5] X.-H. Sun and D. Wang, "APC: A Performance Metric of Memory Systems", ACM SIGMETRICS Performance Evaluation Review, Volume 40, Issue 2, 2012.

[6] Hans de Vries, "Understanding the detailed Architecture of AMD's 64 bit Core", http://chip-architect.com/news/2003_09_21_Detailed_Architecture_of_AMDs_64bit_Core.html, September, 2003.

[7] Intel, Reference Manual, "Introduction to Microarchitectural Optimization for Itanium® 2 Processors", http://developer.intel.com, 2011, April

[8] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt. The M5 simulator: Modeling networked systems. IEEE Micro, 26(4):52{60, July-Aug. 2006.

[9] C. D. Spradling. SPEC CPU2006 benchmark tools. ACM SIGARCH Computer Architecture News, 2007.

[10] D. Wang and X.-H. Sun, "Concurrent Average Memory Access Time", Illinois Institute of Technology Technical Report (IIT/CS-SCS-2012-05), 2012.

Xian-He Sun is a Professor of Computer Science and the Chairman of the Department of Computer Science at the Illinois Institute of Technology (IIT). He is the director of the Scalable Computing Software laboratory at IIT, an IEEE fellow, and is a guest faculty in the Mathematics and Computer Science Division at the Argonne National Laboratory. Before joining IIT, he worked at DoE Ames National Laboratory, at ICASE, NASA Langley Research Center, and at Louisiana State University, Baton Rouge. Dr. Sun's research interests include parallel and distributed processing, memory and I/O systems, software systems, and performance evaluation and optimization.
Contact Information: Xian-He Sun
Mailing Address: Department of Computer Science, Illinois Institute of Technology, 10W 31st street, Chicago, IL 60616, USA
Email: sun@iit.edu
Phone: 1-312-567-5260

Dawei Wang received the Ph.D. degree in Computer Science in the Institute of Computing Technology, Chinese Academy of Sciences in 2009. He worked as a postdoctoral researcher at the Scalable Computing Software laboratory in the Illinois Institute of Technology during 2010 and 2012. He is currently working at Juniper Networks for next-generation High-throughput Ethernet Routers. His research interests include computer architecture, large scale interconnection networks, and architectural simulation & emulation.
Contact Information: Dawei Wang
Mailing Address: 1909 Silva Place, Santa Clara, CA, 95054.
Email: david.albert.wang@gmail.com
Phone: 1-408-705-0786

Digital Object Indentifier 10.1109/MC.2013.227 0018-9162/$26.00 2013 IEEE