ILLINOIS INSTITUTE OF TECHNOLOGY

Scalable Computing Software Laboratory Technical Report

Department of Computer Science

Illinois Institute of Technology

Concurrent Average Memory Access Time

Dawei Wang, Xian-He Sun

{dwang31, sun}@iit.edu

May 2012

Technical Report №. IIT/CS-SCS2012-05

http://www.cs.iit.edu
10 West 31st Street, Chicago, IL 60616

Memory Access Cycle and the Measurement of Memory Systems

Dawei Wang and Xian-He Sun
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA 60616
{dwang31, sun}@iit.edu

ABSTRACT

Concurrency is the driving force increasing peak performance in modern computing systems. However, the effectiveness of memory concurrency is application dependent. It varies largely from application to application and even between implementations. Understanding and utilizing memory systems' performance 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 paper, we propose Concurrent Average Memory Access Time (C-AMAT) as an accurate metric for modern memory systems. C-AMAT has the ability to examine concurrent behavior and provides a quantitative performance measurement at component and layer level of modern memory systems. In this study, the concept and formulation of C-AMAT are introduced. Several processor architectures and cache technologies, such as multiple issue pipeline, non-blocking cache, CMP, etc., are studied to validate the feasibility and validity of the C-AMAT metric. Experiment results confirm that C-AMAT is significantly better than the existing memory performance metrics in characterizing concurrency. C-AMAT matches design choices well, while other traditional memory metrics often mislead in measurement when concurrency is present.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: *design studies, measurement techniques, performance attributes.*

General Terms

Measurement, Performance

Keywords

Memory concurrency; memory metric; memory performance measurement

I. INTRODUCTION

The unbalanced technological advancements in processor and memory over the past thirty years directly lead 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 [3]. Additionally, the IBM Power6 has a 4-cycle L1 cache hit latency, and a L2 cache hit latency of 24 cycles [4]. Last Level Caches (LLC) of modern multi-core processors even exceed one hundred cycles [4]. 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 [5], multi-banked cache [6] and pipelined cache [7] are advanced cache technologies which improve cache hit concurrency; whereas, non-blocking cache [8] is the main technology to improve cache miss concurrency. Processor ILP (Instruction Level Parallelism) technologies, 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 technologies, tens of or even hundreds of cache/memory accesses may coexist in the memory hierarchy at the same time. Thus, a single cache miss becomes irrelevant to 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. Understanding the performance of modern hierarchical memory systems with large access parallelism remains elusive for researchers and practitioners. Recently, Sun and Wang have proposed a new memory performance metric APC (Access Per Cycle) to measure concurrent memory system performance [9]. APC introduces the new concept of measuring concurrency. It is an effective measurement tool, but lacks the formulation representation which designers like to see for quantitative analysis. In this study, we extend AMAT to create Concurrent AMAT, C-AMAT. 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. When the concurrencies are equal to one, C-AMAT is the same as AMAT.

APC has been proven to be the most appropriate performance metric for overall memory systems [9]. By definition, as shown in Section 2, C-AMAT is a reciprocal of APC, which confirms of the importance of C-AMAT in memory performance analysis. In fact, C-AMAT is a complement of APC. APC is designed for measuring the actual performance of a memory system. C-AMAT is designed to provide a component based theoretical analysis. Together they form a measurement and evaluation system which captures the concurrency of modern memory systems.

The introduction of C-AMAT is threefold. First, the definition of C-AMAT is introduced; then the conventional AMAT formula has been extended with hit and miss concurrency factors; finally the extended formula has been mathematically proven to be equal to the definition of C-AMAT.

Since previous research [9] has already proven the accuracy and correctness of APC, which guarantees the accuracy of C-AMAT, this paper will more focus on the measurement factors in the C-AMAT's extended hierarchical formula, especially hit and miss concurrency factors. We will re-examine the impact of the choices of processor microarchitecture and cache structure on cache access concurrency with C-AMAT and existing memory performance metrics. Experimental results show that only C-AMAT always matches the design choices of modern commercial processors, and existing conventional memory metrics mislead the designer.

The rest of this paper is organized as follows. Section 2 defines C-AMAT, extends the traditional AMAT's formula with hit and miss concurrency factors, and proves the extended formula is equal to C-AMAT. Section 3 discusses the measurement of the factors in the extended formula. Section 4 explains the impact of some key microarchitecture features on concurrency factors. Section 5 provides our experimental methodology. Section 6 presents the experimental results. Section 7 describes related works and, finally, Section 8 summarizes the contributions and conclusions of this paper.

II. DEFINITION AND FORMULATION

The Concurrent Average Memory Access Time (C-AMAT) extends the traditional AMAT with concurrency factors. In this section, the definition of C-AMAT is firstly introduced. Next, the relationship of C-AMAT and APC is deduced based on their definitions. The traditional AMAT formula is then extended with two concurrency factors. Finally, the extended formula is theoretically proven to be equal to the value of C-AMAT. Thus, C-AMAT has a quantitative component-based formula which is more suitable for memory performance analyses.

A. Definition

C-AMAT is defined as the average memory access time considering the overlap between multiple 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

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

The definition of C-AMAT is simply enough. However, due to the advanced structures of the 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. When counting the memory access cycles, the *overlapping* mode is adopted. The overlapped mode means when there are several memory accesses co-existing during the same cycle, but $T_{MemCycle}$ only increases by one. Another important feature of $T_{MemCycle}$ is that $T_{MemCycle}$ *only* includes the clock cycles with cache access activities; the cycles without memory references are excluded. According to the definition of APC (Access Per Cycle) [9], the C-AMAT is the reciprocal of APC. Based on the measurement methodology of APC proposed in [9], the final value of C-AMAT can be obtained.

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

Paper [9] used the statistical variable *correlate coefficient* to verify the correctness and accuracy of different memory metrics, including MR, AMP, and AMAT. The simulation results of [9] indicated that APC is the most accurate memory metric to reflect the overall memory system performance. The direct relation of C-AMAT and APC translates the correctness and accuracy of APC to C-AMAT. However, like AMAT, C-AMAT is designed to signify component-wide performance analysis. A component-based, parameterized formula of C-AMAT needs to be derived in order to put C-AMAT in use.

B. 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. The deficiency of AMAT is that there are no factors to describe the concurrency of memory accesses, in either the hit part or the miss part of the formula. AMAT assumes the memory accesses are sequential, one after another. This point of view may be correct twenty years ago, but does not fit modern processor architectures or memory hierarchies. Furthermore, AMAT ignores the relation between hits and misses existing at the same cycle. In the following section, we extend the AMAT formula by introducing concurrency parameters for the hit and miss accesses, and propose a new counting method for MR and AMP which considers the relation between concurrent hits and misses. With these modifications, the extended formula is then theoretically proven to be equal to formula (1). The extended formula is shown as below.

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

The first parameter C_H represents the hit concurrency; the second parameter C_M represents the miss concurrency. The C_H can be contributed by multi-port cache, multi-banked cache or pipelined cache structures. The C_M can be contributed by the non-blocking cache

structure. In addition, processor ILP technologies, 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 the miss contains at least one miss cycle which does not have any hit access activity. When measuring private caches for CMP processors, the pure misses is measured based on "per-core" mode. That is every core has its own detecting logic. The related monitor logic only detects this core's private cache hit accesses when this core has private cache misses. If no hit access, then the correspondent cycles are measured as "pure miss cycles" for this core. Also, for shared caches, the pure miss cycles is measured based on "all-core" mode. That is a cycle is called a "pure miss cycle" if there is no any cache hit access on any core when a cache miss occurs. AMP is also re-defined as the average number of pure miss cycles per miss access. Fig. 1 illustrates a concrete example.

There are 5 different memory accesses in Fig. 1. Each access contains 3 cycles for cache hit operations. If it is a miss, extra miss penalty cycles will be appended, and the number of miss penalty cycles is uncertain. Access 1, 2, and 5 are hit accesses; Access 3 and 4 are miss accesses. Access 3 has a 3-cycle miss penalty; Access 4 has only a 1-cycle miss penalty. As we can see, when considering the access concurrency, only Access 3 contains 2 pure miss cycles. Even though, Access 4 has 1 miss cycle, this cycle cannot be counted as a pure miss cycle, because this miss cycle of Access 4 is overlapped with Access 5 hit cycles. According to the new definition of miss rate, the miss rate of the five accesses is 0.2 instead of 0.4 for the non-concurrent version. The reason for omitting the misses whose cycles are totally overlapped with hit accesses is that this kind of miss accesses will not drag down processor performance, since the processor can continue generating memory accesses while waiting for the missing data returning from lower memory hierarchies. According to formula (1), C-AMAT is 8 cycles/ 5 accesses = 1.6 cycle per access; whereas AMAT is 3+0.4×2=3.8 cycle per access. From the processors point of view, the processor will get one missing data every 1.6 cycles, not 3.8 cycles.

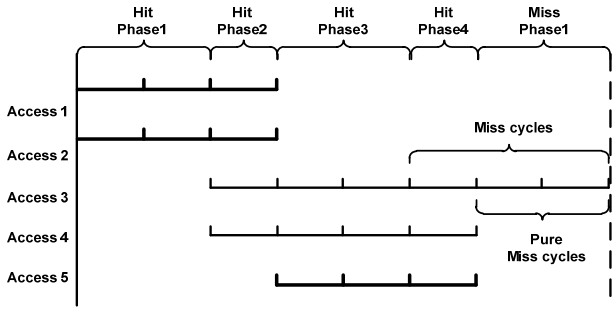

Figure 1. A C-AMAT Example

Another approach to calculate C-AMAT is to use hit and miss concurrency factors. 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 C_i will not change; C_i is the hit concurrency during the phase i; t_i is the continuous lasting cycles for 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 inside 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 C_j will not change; C_j is the miss concurrency during the phase *j*; t_j is the continuous lasting cycles for phase *j*. Note the pure miss phases only includes 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 cannot be counted in pure miss phases. T_M is the total pure miss cycles in the overlapping mode, therefore

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

For the example as shown in Fig. 1, there are 4 hit phases, namely Hit phase 1, 2, 3, 4, which contain 2, 4, 3, 1 concurrent hit cache assesses with lasting cycle 2, 1, 2, 1 respectively. Therefore, $C_H = 2\times2/6 + 4\times1/6 + 3\times2/6 + 1\times1/6 = 5/2$. And there is only one pure miss phase with 1 pure miss concurrency which lasts for 2 cycles. Therefore $C_M = 1\times2/2 = 1$; AMP = 2/1 =2; MR = 1/5. Thus formula (3) is equal to

$$\frac{H}{C_H} + MR \times \frac{AMP}{C_M} = \frac{3}{5/2} + \frac{1}{5} \times \frac{2}{1} = 1.6$$

So, according to the newly defined MR and AMP based on pure miss concept, formula (1) and (3) yield the same result. In the next section, we will theoretically prove formula (1) is equal formula (3) based on their definition. More interestingly, formula (3) gives more information to improve cache performance than traditional AMAT formula. Also formula (3) can be iteratively extended from L1 cache down to LLC.

C. 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.

MR (Miss Rate) in this study is different from the traditional miss rate definition. Only when a miss access has non-overlapping cycles with all 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

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

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

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 we have

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

According to formula (5), there are five important factors determining the overall memory performance, namely hit latency, hit concurrency, miss rate, average miss penalty, and miss concurrency. Even though the concepts of miss rate and average miss penalty in C-AMAT are similar with the counterparts used in AMAT, C-AMAT excludes hit and miss accesses overlapping cycles. In the following sections, we will demonstrate the importance of hit concurrency and miss concurrency. Without concurrency, the traditional memory metrics may mislead the design considerations. Only with consideration of concurrency, the memory performance can reflect the impact of micro-architectural design choices correctly. Also we will see in section VI, different processor and cache technologies will affect different C-AMAT parameters, sometimes might improve one factor while deteriorate other parameters. Only comprehensively considering all factors can represent the correct memory performance.

III. CACHE CONCURRENCY MEASUREMENT

As introduced in formula (5), there are 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. No matter a cache access finally turns out to be a hit or a miss access, this cache access has to spend a certain fixed cycles in the cache tags query. With considering advanced cache technologies, 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 [10], thus the maximum hit concurrency is 2×3=6. The miss concurrency 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 factors' measurements, the difference of that of C-AMAT is that miss cycles and the number of misses do not include the hit and miss overlapping parts. The reason is that, when there are overlaps between hit and miss accesses, the memory does not block CPU performance; CPU is able to continue running with enough operand-ready instructions. Only the pure miss access cycles slow down the CPU speed. Thus, the difficulty of measuring the miss concurrency is to eliminate the overlapping cycles between hit accesses and miss accesses. That is to say, the miss concurrency detector needs to simultaneously aware cache hit accesses and miss accesses. Only the cycles in which at least one miss access exists and no hit accesses will be counted as one pure miss cycle. If one miss contributes at least one pure miss cycle, then this miss will be counted as one pure miss. The detailed cache hit and miss concurrency detecting structure is shown in Fig. 2.

The Hit Concurrency Detector (HCD) counts the total hit cycles and record 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 cycles which do not include any hit access activities will not be counted as hit cycles. Also HCD tells Miss Concurrency Detector (MCD) whether current cycle has hit accesses or not. MCD is the monitor unit to count the total pure miss cycles and record each pure miss phase in order to calculate the average miss concurrency, pure miss rate and pure miss penalty. With the information provided by HCD, MCD is able to tell whether a cycle is pure miss cycle or not, and whether a miss is pure miss or not. 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 at component level.

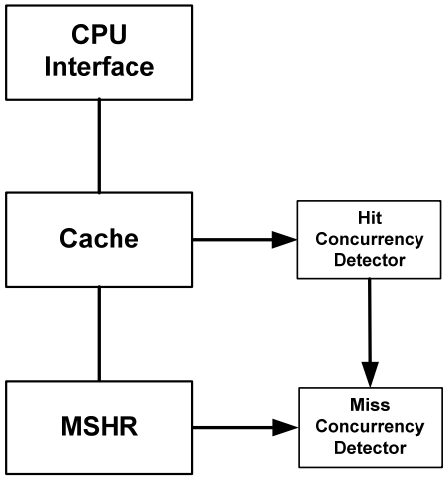

Figure 2. C-AMAT Detecting Structure

IV. MICROARCHITECTURE IMPACT

In this section, several processor microarchitectures and advanced cache technologies are presented in order to demonstrate their contributions on cache concurrency. From processor microarchitecture aspect, several critical ILP technologies or functional units, such as out-of-order execution, multiple issue pipeline, load/store unit, issue buffer and reorder buffer, not only has large impact on instruction parallelism, but also on cache/memory access concurrency. SMT can execute multiple threads' instructions in one cycle; while CMP can execute multiple independent applications concurrently within different processor cores on the same die. Both of these technologies can

also dramatically increase cache hit and miss concurrency. Advanced cache technologies, such as multi-port cache, multi-banked cache, pipelined cache, can directly determine the maximum cache hit concurrency; other cache advanced technologies, such as non-blocking cache, cache prefetching, main memory bank-level parallelism, greatly influence the cache miss concurrency.

In the sake of space limitation, in this paper, we only select five basic and fundamental technologies to verify the accuracy of C-AMAT. The five technologies or function units are multiple issue pipeline, issue buffer and re-order buffer, cache size and associativity, non-blocking cache, and chip multi-processing.

A. Multiple Issue Pipeline

Multiple issue pipeline allows multiple instructions to be fetched, decoded, issued, executed and committed at the same cycle. It is a very important ILP technology widely used in modern commercial processors. Due to data dependency of algorithms, high penalty of branch mis-prediction, missing load data, and "power wall" problem, over-increasing pipeline width will not improve application performance much. Usually the commercial processors adopt 4~8 width pipeline at different stages [3] [11] [6]. In this paper, we assume each stage has the same width for simplicity.

B. Issue Buffer and Re-Order Buffer (ROB)

Out-of-order processors usually adopt issue buffer and ROB as hardware function unit to implement out-of-order execution and in-order retirement mechanism. Issue buffer (also known as reservation station) and ROB size directly determine the maximum number of in-flight instructions, thus can greatly influence the cache access concurrency. The larger the two buffer sizes are, the higher cache access concurrency could be. However, the concurrency is also influenced by the load data return place. When a missing load instruction becomes the oldest entry in the ROB, it will hold up further instruction retirement until its expected data returns. When the miss load data is found in LLC or main memory, ROB and issue buffer could be become full very soon. As a consequence, no more instruction can be issued, and no more loads or stores can be generated, the cache/memory concurrency is compromised. Therefore, both the issue buffer size and the ROB size have a potentially large impact on the data access concurrency. In this paper, we assume issue buffer and ROB have the same size for the sake of discussion.

C. Cache Size and Associativity

Cache size and associativity are the most important and fundamental cache configurations. They can directly influence cache miss rate, and also indirectly affect current and next level cache concurrency. The more data founded in current cache level, the faster processor could execute instructions, and then the higher cache concurrency is. Usually lower miss rate in current level will result in lower concurrency in next cache level.

D. 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 [12], Itanium [13], and IBM POWER3 [14], employ non-blocking cache heavily at each level of a memory hierarchy in order to enhance memory access parallelism. Non-blocking cache can continue supplying data under certain number of cache misses by adopting Miss Status Holding Register (MSHR) [8]. MSHR is a structured table. It 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, designates there is no outstanding main memory access. When the MSHR table is full, then the cache cannot afford more cache accesses, and the CPU's memory accesses or next-level memory accesses are blocked due to the lack of MSHR entry. Therefore the number of MSHR entries can directly determine the miss access concurrency.

E. Chip Multi-Processor

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

V. EXPERIMENT SETUP

A detailed out-of-order CPU model in the M5 simulator [15] was adopted, which models an Alpha 21264-like CPU. Unless stated otherwise, the experiments assume the following default processor and cache configuration showing in Table I.

TABLE I. TABLE I. DEFAULT SIMULATION CONFIGURATION PARAMETERS

Parameter	**Value**
Processor Function units ROB, LSQ size	1core, 4 GHz, 4-issue width, 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 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 paper, there are five different sets of configurations based on the default configuration. Each of them only changes one or two parameter/s of the simulation in order to show each architecture technology influence on the cache concurrency and C-AMAT. The five technologies, as stated in Section IV, are multiple issue pipeline, issue buffer and ROB size, non-blocking cache, cache size and associativity, and multi-core technology. The variation trends of all memory performance metrics, including MR, AMP, AMAT and C-AMAT could distinguish the correctness of each memory performance metrics. The winner should always correlate the processor design choices well. The detailed experiment configurations are shown in Table II.

TABLE II. TABLE II. A SERIAL OF SIMULATION CONFIGURATIONS

Configuration Set	Description
Set 1	Change **multiple issue pipeline** width with value of 1→2→4→8
Set 2	Change **issue buffer size and ROB size** with value of 16→32→48→64→80→96→128
Set 3	Change **MSHR entry size** with value of 1→2→4→8→16
Set 4	Change **cache size and associativity** L1 cache size: 16KB→32KB→64KB L1 cache associativity: 2→4→8
Set 5	Change **core number** with value of 1→2→4→8; Also L2 cache size changes according to core number with value of 512KB→1MB→2MB→4MB; Each core runs the same benchmark.

The simulations were conducted with 24 benchmarks from SPEC CPU2006 suite [16]. 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, 10M instructions were simulated to collect statistics. Each value of memory performance metrics represented in section VI is the average value of all 24 benchmarks.

VI. EXPERIMENT RESULTS

In this section, we show the results of the experiments listed in Section V to verify C-AMAT's capability in measurement. By comparing results of C-AMAT and AMAT for each technology, one can clear see that only C-AMAT always matches actual design choices of modern processors. Science the emphasis here in is on the comparison of C-AMAT and AMAT, not the performance of the benchmarks, only the average values of the correspondent memory metrics of the 24 benchmarks are shown in the following figures.

A. Impact of Multiple Issue Pipeline Width

Multiple issue pipeline is one important processor microarchitecture technology to improve ILP. Usually, modern general purpose CPUs use 4~8 issue width. From Figure 3 (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 cannot correctly reflect the memory performance variation trend.

Fig. 3(a) shows the average hit concurrency, and average pure miss concurrency of L1 data cache when issue width increasing from 1 to 8. It can be observed that the average hit concurrency continue increase, whereas average pure miss concurrency increases slowly when pipeline width larger than 4. According to Fig. 3(b), the traditional AMAT increases when the pipeline width increases continually, which means the memory performance is decreasing. This is a contradiction with the fact that application performance is improved with the increased width of issue pipeline width [6]. On the contrary, C-AMAT with consideration of hit and miss concurrency, not only correctly describes that the overall memory performance is increasing, but also correctly reflects that the performance speedup decreases with the issue width increases when the pipeline issue width are larger than 4. It is interesting to note that this C-AMAT result confirms that the current choice of 4~6 issue width of general purpose processors is an optimal design choice. That demonstrates the practical usefulness of C_AMAT. According to Fig. 3(c) and Fig. 3(d), the MR and AMP cannot correctly the memory performance either. Only the comprehensive memory metric C-AMAT which considers hit and miss access delay, proportion and concurrency can correctly reflect the overall memory performance.

It also should note that the difference between MR and Pure MR, AMP and Pure AMP, are very small. For miss rate the average difference is 1.2%, the value for miss penalty is 7.1%. More importantly, the variation trends of the two parameters in each pair are almost identical. That is to say, for the configuration of changing pipeline issue width, the correctness of C-AMAT is mainly determined by the concurrency of memory accesses. Therefore, we can conclude that concurrency is the critical performance factor to reflect the performance of modern memory systems, which was not a critical issue when AMAT was first introduced years ago.

In the following sections, we can observe that different architecture technologies have different impacts on the five parameters of the C-AMAT's equation. Ignoring any of them will misunderstand the overall memory performance. C-AMAT considers all of these five memory metrics. It is the most appropriate metric to reflect the overall memory performance.

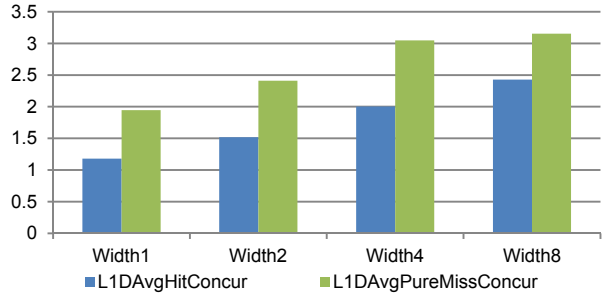

Figure 3(a). L1 DCache Concurrency when Changing Issue Pipeline Width

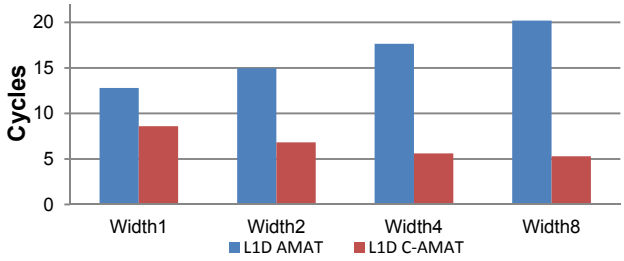

Figure 3(b). L1 DCache AMAT and C-AMAT when Changing Issue Pipeline Width

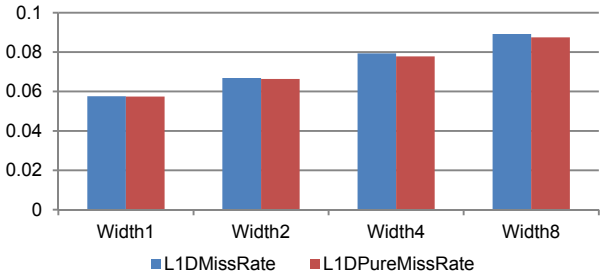

Figure 3(c). L1 DCache Miss Rate and Pure Miss Rate when Changing Issue Pipeline Width

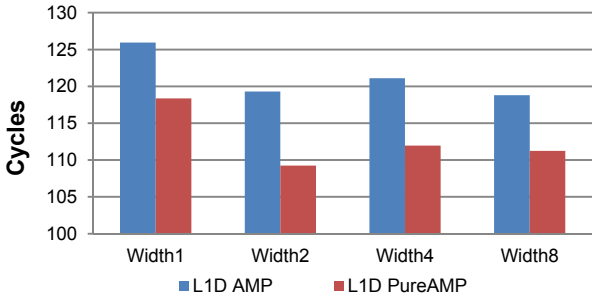

Figure 3(d). L1 DCache AMP and Pure AMP when Changing Issue Pipeline Width

B. Impact of Issue Buffer Size and ROB Size

The size of issue buffer and ROB buffer directly determine the maximum number of in-flight instructions. However, oversized ROBs cannot exploit more data access concurrency, due to data dependency, data reference instructions' proportion, and current location of missing data. From Fig. 4(a), it can be seen when the issue buffer size and ROB size are equal to or larger than 64, the all hit and miss concurrency have negligible increasing. That is the reason why we choose 64 as the size of issue buffer and ROB in our basic configuration. Also according to Fig. 4(b), AMAT gives wrong memory performance predictions, which predicts the memory system performance decreases with the increase of the size of issue buffer and ROB. On the contrary, C-AMAT reflects the right memory performance variation.

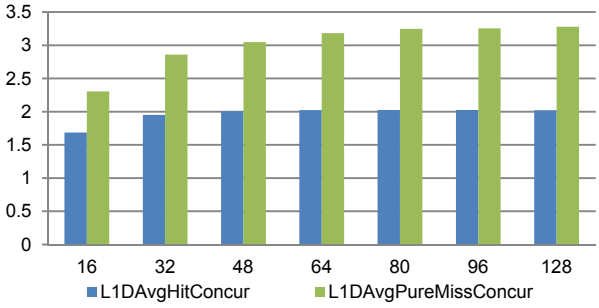

Figure 4(a). L1 DCache Concurrency when Changing Issue Buffer

and ROB Size

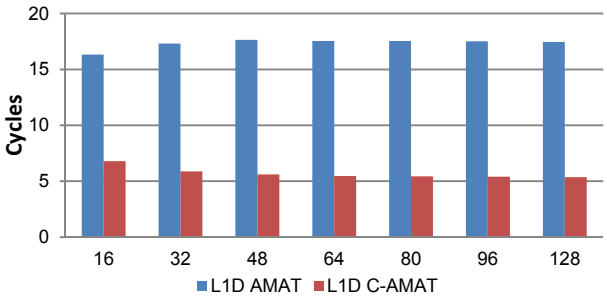

Figure 4(b). L1 DCache AMAT and C-AMAT when Changing Issue Buffer and ROB Size

C. Impact of Cache Configuration

As we previously discussed, the cache configuration will affect the data access latency, and change the data access concurrency as well. From Fig. 5(a) we can observe that the average hit concurrency constantly increases with the increase of the cache size and associativity, even though sometime the improvement is small in proportion. However, for the average pure miss concurrency, there is no noticeable pattern (Fig. 5(b)). According to Fig. 5(c), in which all the pure miss rate values are normalized to 16KB 2-way set associate case (the worst case in this configuration set), the miss rate decreases as much as 33%. It is much larger than concurrency variation (<3%). Therefore, we can conclude that changing the cache configuration (size and associativity) mainly influence the miss rate parameter, not concurrency. When the mainly changing factor of memory performance is miss rate, we can found that the AMAT and C-AMAT have the same variation trends according to Fig. 5(d) and 5(e). It can be observed that for L1 data cache, when the set associativity is larger than 2, the memory performance improvement is very limited for each size category. According to Fig. 5(d) and 5(e), the best performance cost option is 32KB/64KB 4way set associate, which is the common choice for a lot of commercial processors for L1 data cache options [3] [17] [18].

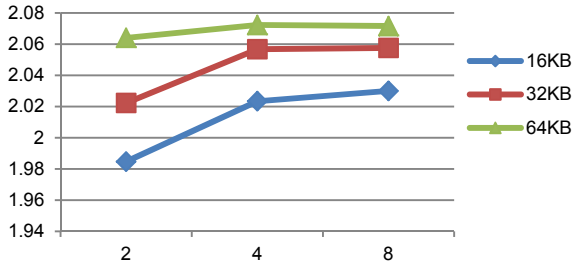

Figure 5(a) L1 DCache Average Hit Concurrency when Changing Cache Configuration

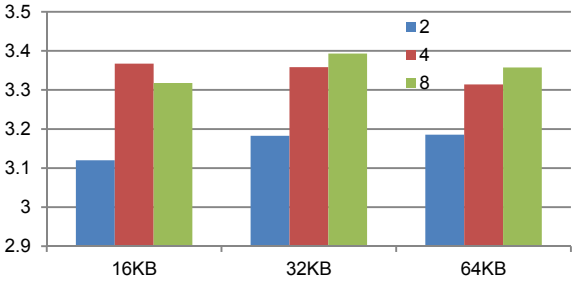

Figure 5(b) L1 DCache Average Pure Miss Concurrency when Changing Cache Configuration

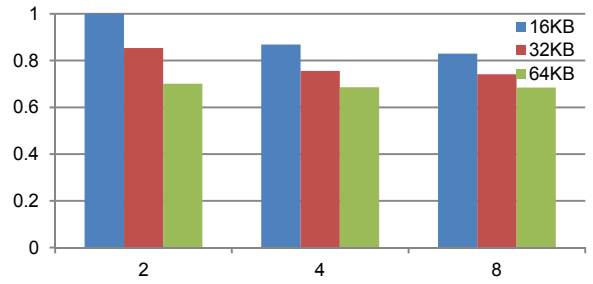

Figure 5(c) L1 DCache Miss Rate when Changing Cache Configuration

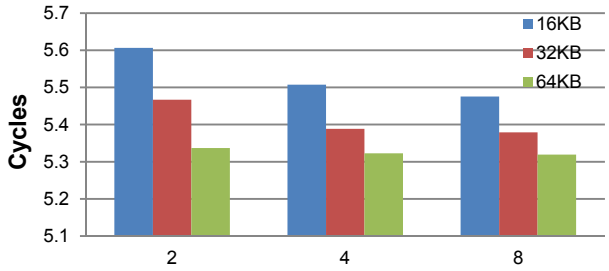

Figure 5(d) L1 DCache C-AMAT when Changing Cache Configuration

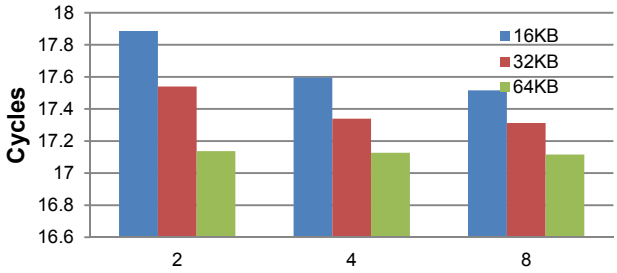

Figure 5(e) L1 DCache AMAT when Changing Cache Configuration

D. Impact of MSHR Size

The size of MSHR table can directly determine the maximum miss concurrency. It does not have direct impact on hit concurrency. As shown in Fig. 6(a), when the number of MSHR entry is increased, the average hit concurrency approximately maintains the same, whereas the average pure miss concurrency constantly increases. But the larger the MSHR table is, the smaller the miss concurrency gain is. According to Fig. 6(b), when the number of MSHR entry increases, AMAT increases, which means the memory performance decreases, and C-AMAT decreases. The latter obviously matches the wide adoption of the non-blocking cache technologies in modern processor design [12] [13] [14]. In other words, AMAT gives a false indication about memory system performance.

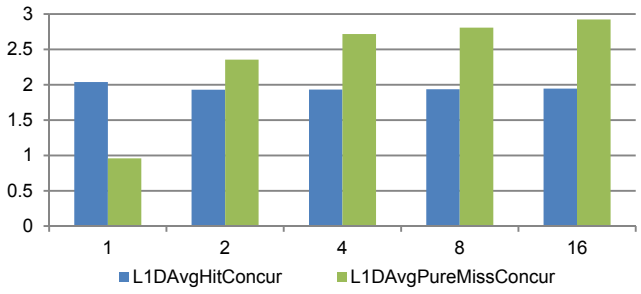

Figure 6(a) L1 DCache Concurrency when Changing MSHR Size

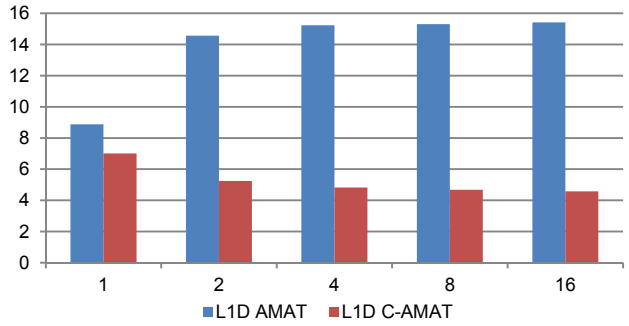

Figure 6(b) L1 DCache AMAT and C-AMAT when Changing MSHR Size

E. 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. According to Fig. 7(a), the L2 cache concurrencies constantly increase with the number of cores. Especially for the two miss concurrencies, they increase very fast. The miss rates are also increased due to the increases of conflicts (see Fig. 7(b)). Without considering concurrency, one may conclude that the overall performance of L2 cache decreases. However, this obviously conflicts with the widespread use of multi-core processors. Only with the concurrency contribution, the overall memory performance of L2 cache which is represented by C-AMAT increases. This observation also reflects the importance of concurrency of data accesses.

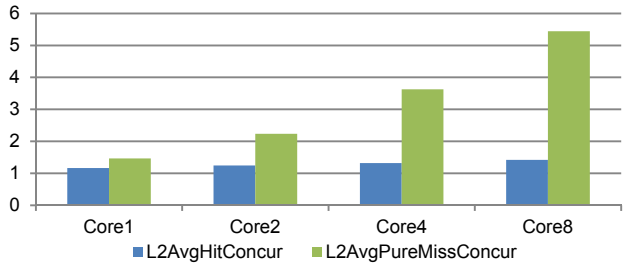

Figure 7(a) L2 Cache Concurrency when Changing Core Number

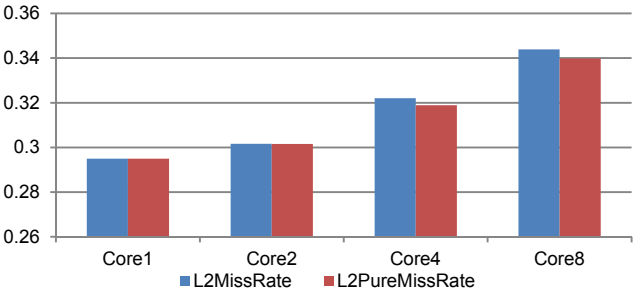

Figure 7(b) L2 Cache Miss Rate when Changing Core Number

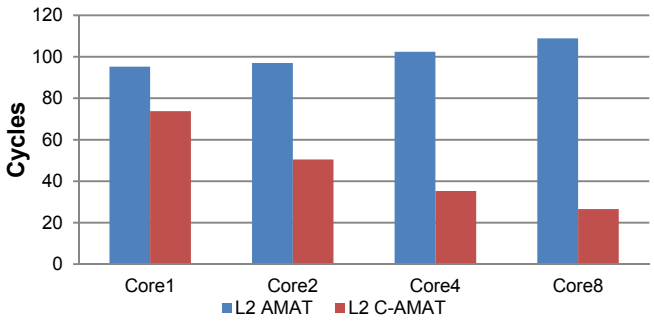

Figure 7(c) L2 Cache AMAT and C-AMAT when Changing Core Number

VII. RELATED WORK

Miss Rate (MR), Average Miss Penalty (AMP), and Average Memory Access Time (AMAT) are commonly used performance metrics in evaluating memory systems [6]. MR is defined as {the number of miss memory accesses} over {the number of total memory accesses}. AMP equals {the summary of single miss latency} over {the number of miss memory accesses}. Finally, AMAT = Hit time + MR×AMP. MR only reflects the proportion of the data in or out of the cache; they don't reflect the penalty of the miss access. AMP only catches the penalty of the cache miss access; it doesn't show the hit in performance. Also for AMP, the single miss latency is counted based on one miss access as if there no other memory access present. AMAT is a comprehensive memory metric, but it is still based on the single data access point viewpoint. It is the average time between start time and finish time of a single memory access. It does not consider the memory hit and miss access overlap and parallelism. Therefore, AMAT cannot reflect the memory system performance as a whole. Memory bandwidth is usually used to measure the peak performance of memory systems with intensive memory workloads. The biggest difference between memory bandwidth and C-AMAT is the way they measuring the time. Memory bandwidth measures the time period including all clock cycles, whereas C- AMAT measures the time period only including the clock cycles with memory access activities. Thus, C-AMAT reflects more accuracy of memory system performance, especially when CPU cycles are not equal to memory cycles.

There are several studies on Memory Level Parallelism (MLP) [19] in recent years. MLP is a main memory concurrency metric. It is the average number of long-latency main memory outstanding accesses when there is at least one such outstanding access [20]. MLP actually is the concurrency metric of main memory access. A known limitation of MLP is it only focuses on off-chip memory access based on the epoch memory access mode for some commercial or database workload [20]. These kinds of applications usually have much more L1, L2 cache misses and their overall performances are heavily determined by main memory access. But for some traditional CPU intensive applications, only consider main memory access is far from enough. In addition, advanced process technologies, such as EDRAM used in IBM POWER7, also dramatically increase on-chip cache size up to 32MB [21]. 3D integration technologies, can implement multi-layer CPU with one layer full of on-chip cache [22]. In contrast, C-AMAT not only can be applied to L1, L2 or main memory level respectively, but also includes concurrency, miss rate, miss penalty parameters. C-AMAT is a comprehensive memory performance metric which can represent the overall memory system performance. That makes C-AMAT useful in analyzing commercial applications and traditional scientific applications as well. C-AMAT has a much wider application spectrum than MLP. Nevertheless, MLP is a new metric drawing much attention for data-intensive applications. Being a superset of MLP demonstrates the preeminence of C-AMAT from another angle.

Recently proposed memory Access Per Cycle (APC) mimics the concept of Instruction Per Cycle (IPC). It fully considers the concurrency inside memory systems. APC has been proven as the most appropriate memory metric to reflect the overall memory performance [9]. But APC only provides final memory performance values, and lacks of component based formulation for hierarchical analysis. C-AMAT complements APC by providing five specific memory performance factors, which includes hit/miss concurrency, hit latency, miss rate and average miss penalty. It is an extension of AMAT and is in easy for anyone familiar with AMAT.

VIII. CONCLUSIONS AND FUTURE WORK

In this study, we have proposed a new memory metric C-AMAT which considers data access concurrency in modern memory hierarchy, have theoretically proved C-AMAT component-based formula, and presented the measurement methodology of C-AMAT. Extensive simulations are conducted to confirm that C-AMAT always matches the modern processor architecture design choices. We have shown C-AMAT is an extension of the widely used conventional AMAT metric. When access concurrency remains the same, AMAT reflects memory improvements well, so does C-AMAT. However, when the memory access improvement comes from concurrence by using ILP or advanced cache technologies, 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 catch the concurrence improvement. Compared with the recently proposed APC metric, C-AMAT provides a more component-based analytical study on memory performance. C-AMAT with its parameterized formula, can give designers more clear understanding about componential influence in the overall memory performance. In the future, we plan to carry out a more detailed study of C-AMAT on different levels of memory hierarchies, and to evaluate more advanced memory technologies, such as hardware prefetching, and also try to extend C-AMAT to evaluate SMT on multi-core environments.

IX. 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] Michael E. Thomadakis, "The Architecture of the Nehalem Processor and Nehalem-EP SMP Platforms", A research report of Texas A&M University, http://sc.tamu.edu/systems/eos/perf_nehalem.pdf

[4] Robert Fiedler, "Blue Waters Architecture", Great lakes consortium for Petascale Computation. October 2010

[5] Jude A. Rivers, Gary S. Tyson, Edward S. Davidson, and Todd M. Austin, "On High-Bandwidth Data Cache Design for Multi-Issue Processors", in Proceeding of the 30th annual ACM/IEEE international symposium on Microarchitecture (MICRO 30), Washington, DC, USA, 1997.

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

[7] Amit Agarwal, Kaushik Roy, T. N. Vijaykumar, "Exploring High Bandwidth Pipelined Cache Architecture for Scaled Technology", in Proceedings of the conference on Design, Automation and Test in Europe(DATE'03), 2003.

[8] 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.

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

[10] 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.

[11] B. Sinharoy, R. Kalla, W. J. Starke, etc, "IBM POWER7 multicore server processor," in IBM Journal of Research and Development , Vol.55, No.3, pp.1-29, May 2011.

[12] Intel White Paper, Inside Intel Core Microarchitecture and Smart Memory Access, http://www.iuma.ulpgc.es/~nunez/procesadoresILP/ Intel64-Core-smartmemoryaccess-sma.pdf, 2011, April

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

[14] Stefan Andersson, Ron Bell, John Hague, et.al, "RS/6000 Scientific and Technical Computing: POWER3 Introduction and Tuning Guide", http://www.redbooks.ibm.com, 2011, April.

[15] 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.

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

[17] H. Q. Le, W. J. Starke, J. S. Fields, et. al., "IBM POWER6 microarchitecture", in IBM Journal of Research and Development, Vol.51, No.6, pp.639-662, November 2007.

[18] C.N. Keltcher, K.J. McGrath, A. Ahmed, P. Conway, "The AMD Opteron processor for multiprocessor servers," in IEEE Micro, Vol.23, No.2, pp. 66-76, March 2003.

[19] A. Glew, "MLP yes! ILP no!," in ASPLOS Wild and Crazy Idea Session '98, October 1998.

[20] Y. Chou, B. Fahs and S. Abraham, "Microarchitecture Optimizations for Exploiting Memory-Level Parallelism", in 31st International Symposium on Computer Architecture(ISCA04), June 2004.

[21] William Starke, "POWER7: IBM's Next Generation Balanced POWER Server Chip", in hotchip 21, 2009

[22] Yuh-Fang Tsai, Feng Wang, Yuan Xie, "Design Space Exploration for 3-D Cache", in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, page 444 - 455, 2008