C^2-Bound: A Capacity and Concurrency Driven Analytical Model for Many-core Design

Yu-Hang Liu, Xian-He Sun
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
{yuhang.liu, sun}@iit.edu

***Abstract*—In this paper, we propose C^2-Bound, a data-driven analytical model, that incorporates both memory capacity and data access concurrency factors to optimize many-core design. C^2-Bound is characterized by combining the newly proposed latency model, concurrent average memory access time (C-AMAT), with the well-known memory-bounded speedup model (Sun-Ni's law) to facilitate computing tasks. Compared to traditional chip designs that lack the notion of memory concurrency and memory capacity, C^2-Bound model finds memory bound factors significantly impact the optimal number of cores as well as their optimal silicon area allocations, especially for data-intensive applications with a none parallelizable sequential portion. Therefore, our model is valuable to the design of new generation many-core architectures that target big data processing, where working sets are usually larger than conventional scientific computing. These findings are evidenced by our detailed simulations, which show with C^2-Bound the design space can be narrowed down significantly up to four orders of magnitude. C^2-Bound analytic results can be either used in reconfigurable hardware environments or, by software designers, applied to scheduling, partitioning, and allocating resources among diverse applications.**

***Keywords*-Memory wall; data stall time; memory bound; data access concurrency; Sun-Ni's Law; chip design; concurrent average memory access time (C-AMAT)**

I. Introduction

An on-chip multiprocessor (CMP) is an integrated circuit that consists of two or more independent actual processing units (called "cores") to read and execute program instructions. The design space exploration of on-chip multiprocessors is to investigate potential configurations of integrated circuits with diverse goals, including enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks. Due to these merits, on-chip multiprocessors have become the mainstream of microprocessors that underpin the pivotal computing infrastructure [1]. In the meantime, the amount of cores is continuously increasing on processors. The continual increase is caused by the need to exploit parallelism for different applications whose behaviors require adaptive and optimal on-chip area allocation of cores, caches, and memory controllers. This improvement imposes additional challenge to the already extremely huge design space of CMP that is composed of intractable combinations of a large number of architecture parameters for optimization [2].

A variety of analytical models have been presented recently [3]–[8] to address these challenges. For example, Cassidy and Andreou incorporate sequential data access delay in terms of average memory access time (AMAT) into Amdahl's law [3] [4] [5], while Hill and Marty apply Amdahl's concepts to multi-core architectures based on a hardware cost model [6]. Woo and Lee follow up with the consideration of energy efficiency [7]. In contrast, Sun and Chen consider the impact of memory capacity for many-core design, but they do not explicitly incorporate data access concurrency in their analysis [8]. Although these works, more or less, optimize the design space of CMP in different forms, few of them explore the memory concurrency. As a consequence, data access patterns in these studies are exploited from sequential perspective using the AMAT metric, which cannot truly reflect current reality. In addition, most of these studies are also short in the consideration of memory capacity impact on problem sizes as well, which is another important factor in the design space of CMP.

As data access delay is dominating the overhead in modern big-data processing, it has become the most preeminent performance bottleneck of computing systems. For example, the processor stall time due to data access typically contributes 50% to 70% of the total application execution time [9] [10]. As such, incorporating data access patterns into performance models becomes vitally important. In fact, memory concurrency as the main form of data access patterns has become a more prevalent factor to the design of efficient memory systems (e.g. multi-port, multi-banked, pipelined cache, non-blocking cache, runahead, and simultaneous multithread). Meanwhile, in addition to data concurrency, in scalable computing problem size increases with computing resources and the increase is usually bounded by the available memory size [11] [12] [13]. Hence, assuming the problem size is fixed would cause misleading results. As a result, it is urgently needed to include memory

SC'15, November 15-20, 2015, Austin, TX, USA
 ISBN 978-1-4503-3723-6/15/11 $15.00
DOI: http://dx.doi.org/10.1145/2807591.2807641

concurrency and memory capacity considerations in the many-core design space exploration (DSE) to keep up with the increasing importance of modern memory systems and with the emerging of data intensive applications. To the best of our knowledge, we propose for the first time to simultaneously consider data access concurrency and memory capacity to explore the design space of CMP.

Memory wall and memory bound are two well-known performance constraints [11] [14]. C-AMAT is a new model that unifies the combined impact of data locality and concurrency on data access [15] [16]. Thus applying C-AMAT to many-core design is a natural choice. In the meantime, Sun-Ni's law is a generalization of Amdahl's law and Gustafson's law [11] [17] [18]. While these scalable laws are well studied, they are traditionally discussed in the context of supercomputing. Therefore, applying Sun-Ni's law for many-core processor design, while challenging, has practical significance. In this paper we present C^2-Bound, a data-driven analytical model that incorporates both memory concurrency and memory capacity factors for many-core design. The essence of this model is to take advantage of both the C-AMAT and Sun-Ni's law to optimize the design space of CMP. In particular, this study makes the following contributions:

- C^2-Bound model is proposed to consider both memory locality and concurrency at the same time. To this end, we derive program-specific model parameters from traces and consider adaptively reshaping (through allocating and scheduling) the underlying architecture.
- C^2-Bound model considers problem sizes that are bounded by variable memory capacities. When the number of cores, N, is changed, the on-chip cache capacity will be adjusted correspondingly, and then the problem size will be scaled to a different value. We termed it as the problem size scale function $g(N)$ and found $g(N)$ is a vital factor in balancing the number of cores and the size of the caches. When $g(N) < O(N)$, few cores but large caches are needed; when $g(N) \geq O(N)$, more cores and smaller caches are preferred. These results can significantly narrow the large design space and they can only be got from analytical model rather than from an architect's intuition.
- Finally, C^2-Bound model presents an interface between analysis and simulation. With the APS (Analysis plus Simulation) algorithm, the newly proposed DSE model has been integrated into the GEM5 simulator [19] to supervise simulation. The APS approach drastically reduces the number of required simulations. Representative results for diverse applications confirm the feasibility and correctness of the newly proposed analytical model for many-core processor design. C^2-Bound model has been implemented as an automatic tool to find an application-specific optimal architecture.

The remainder of this paper is organized as follows. The next section provides some preliminary knowledge of C-AMAT and Sun-Ni's law. Section III then proposes the data-driven C^2-Bound analytical model for many-core design. Section IV presents application specific design exploration case studies. Section V further discusses memory concurrency and memory capacity-bounded problem size. Section VI reviews related work in many-core design exploration. Finally, Section V concludes this study and discusses potential future work.

II. Memory Bounds in Terms of Latency and Capacity

Latency and capacity are two bounds of memory on the achievable computing performance. C-AMAT, a new performance metric, accounts for concurrency at both the component and system levels for modern memory design [15] [16] [20]. C-AMAT represents measures and analyzes data access delay from a single program perspective. In contrast, Sun-Ni's law highlights the impact of memory bounded problem sizes on parallel speedup [11]. Both Amdahl's law [17] and Gustafson's law [18] are the special cases of Sun-Ni's law. In this paper, unless otherwise stated, the term memory system indicates the whole memory hierarchy rather than only the main memory.

A. C-AMAT

The conventional memory metric AMAT formulation is shown in Eq. (1) [21], where H is the hit time of data accesses, MR is the miss rate, and AMP is the average miss penalty. AMP is the sum of all miss access latencies divided by the total number of misses. AMAT does not consider the concurrency of data accesses in terms of either hits or misses, based on the assumption that data accesses are sequential, one after another; further, AMAT does not take into account that with concurrent accesses, hits and misses may coexist in the same cycle. The sequential assumption governing AMAT worked well in the past, but applies less accurately for modern processor architectures and memory systems where concurrency is paramount. For example, in an out-of-order processor, when a miss occurs, other instructions can be executed while the memory system is servicing the miss. Moreover, concurrency features such as multi-port, multi-bank and multi-rank allow multiple outstanding reads and writes to co-exist at a given time in the memory system, depending on the underlying hardware support. Therefore, some of the data access latencies can be hidden.

$$AMAT = H + MR \times AMP \tag{1}$$

To cover the concurrent read and write properties of modern memory systems, the C-AMAT model is proposed in Eq. (2) [15]. The first parameter H is the same as that in AMAT. The second parameter C_H represents hit concurrency; the third parameter C_M represents the pure miss concurrency. C_H can be contributed by caches with multi-port, multi-bank or pipelined structures. C_M can be contributed by non-blocking cache structures. In addition, out-of-order execution, multi-issue pipeline, multi-threading and chip multiprocessor (CMP) can all increase C_H and C_M. The pure miss rate, pMR, is different from the conventional miss rate (MR). pMR is the

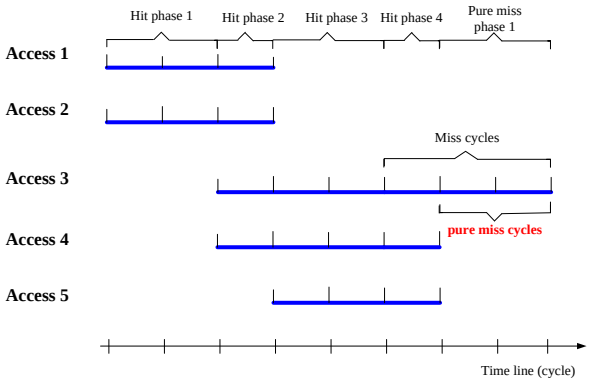

Fig. 1. A demo of C-AMAT and pure miss

ratio of the number of pure (rather than conventional) misses over the total number of accesses. A pure miss here means that a miss contains at least one miss cycle which does not have any hit access activity [15]. pAMP is the average number of pure miss cycles per miss access.

$$C\text{-}AMAT = \frac{H}{C_H} + pMR \times \frac{pAMP}{C_M} \tag{2}$$

As shown in Eq. (3), the ratio of AMAT and C-AMAT is the data access concurrency, which will be abbreviated to C.

$$C = \frac{AMAT}{C\text{-}AMAT} \tag{3}$$

Generally, C is greater than or equal to one. When $C = 1$, we can say there exists no concurrency. At this time, $C_H = 1$, $C_M = 1$, pMR = MR, and pAMP = AMP. Therefore, AMAT can be seen as a special case of C-AMAT. In our later discussion, we will use Eq. (3) to denote the data access concurrency.

Fig. 1 demonstrates C-AMAT concept. There are five different memory accesses and each access contains 3 cycles for cache hit operations. If it is a miss, additional miss penalty cycles will be required. The number of miss penalty cycles is uncertain, depending on where the missed data can be obtained and contention impact during the data access. 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. When considering the access concurrency, only Access 3 contains 2 pure miss cycles. Though Access 4 has 1 miss cycle, this cycle is not a pure miss cycle because it overlaps with the hit cycles of Access 5. Therefore according to our new definition of concurrent (pure) miss rate, the (pure) miss rate of the five accesses is 0.2, instead of 0.4 as that of the conventional non-concurrent version. When miss cycles are overlapping with hit accesses, the processor will not stall; the processor can continue processing the data provided by the hit accesses. According to Eq. (2), C-AMAT is 8 cycles out of 5 accesses or 1.6 cycles per access; whereas by Eq. (1) AMAT is $3 + 0.4 \times 2$ or 3.8 cycles per access. The difference between C-AMAT and AMAT is the contribution of concurrent data access. In this example, concurrency has doubled memory performance.

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, $C_H = 2 \times 2/6 + 4 \times 1/6 + 3 \times 2/6 + 1 \times 1/6 = 5/2$. And there is only one pure miss phase with 1 pure miss concurrency which lasts for 2 cycles. Therefore $C_M = 1 \times 2/2 = 1$; pAMP = 2/1 =2; pMR = 1/5. Thus formula (2) is equal to

$$C\text{-}AMAT = \frac{H}{C_H} + pMR \times \frac{pAMP}{C_M} = \frac{3}{5/2} + \frac{1}{5} \times \frac{2}{1} = 1.6$$

The value of the parameters is in performance analysis and optimization. The invaluable contribution of C-AMAT is that it provides a unified formulation to capture the joint performance impact of locality and concurrency.

B. Sun-Ni's Law

Realizing that the problem size may be constrained by memory capacity, Sun-Ni's law was proposed [11]. Assume each computing node is a processor-memory pair. Increasing the number of processors, N, then, will increase the memory capacity as well. Assume $y = h(x)$ is the relationship between problem size and memory capacity size. That is,

$$W = h(M) \text{ and } W' = h(N \times M)$$

Where M is the memory capacity of one node, W is the original problem size, and W' is the scaled problem size.

Let $y = g(N)$ be the function that reflects the parallel problem increase factor as the memory capacity increases N times (in next subsection, we will present detailed examples to illustrate $g(N)$). By definition,

$$g(N) = W'/W$$

Then we have

$$g(N) = h(N \times M)/h(M) = h(N \times h^{-1}(W))/h(M)$$

Thus memory capacity-bounded speedup is

$$Speedup_{Sun\text{-}Ni} = \frac{f_{seq} \times W + (1 - f_{seq}) \times h(N \times h^{-1}(W))}{f_{seq} \times W + \frac{(1-f_{seq}) \times h(N \times h^{-1}(W))}{N}}$$

f_{seq} is the sequential portion of the problem size. Note that for any power function $h(x) = ax^b$ and for any rational numbers a and b [11], we have

$$h(N \times x) = a(N \times x)^b = N^b \times ax^b = N^b \times h(x) = g(N) \times h(x)$$

Therefore,

$$Speedup_{Sun\text{-}Ni} = \frac{f_{seq} \times W + (1 - f_{seq}) \times g(N) \times W}{f_{seq} \times W + \frac{(1-f_{seq}) \times g(N) \times W}{N}}$$

That is, Eq. (4) holds.

$$Speedup_{Sun\text{-}Ni} = \frac{f_{seq} + (1 - f_{seq}) \times g(N)}{f_{seq} + \frac{(1-f_{seq}) \times g(N)}{N}} \quad (4)$$

Where $g(1) = 1$. Taking $g(N) = N^{3/2}$ as an example, the speedup is

$$Speedup_{Sun\text{-}Ni} = \frac{f_{seq} + (1 - f_{seq}) \times N^{3/2}}{f_{seq} + (1 - f_{seq}) \times N^{1/2}} = O(N)$$

When $g(N) = 1$, Eq. (4) is the Amdahl's law. When $g(N) = N$, Eq. (4) is the Gustafson's law. Because Amdahl's law [17] as well as Gustafson's law [18] both can be seen as the special cases of Sun-Ni's law [11], we will use Sun-Ni's law in Eq. (4) as the base for our further discussion.

For a better understanding, we use some examples to illustrate the role of $g(N)$. Dense matrix multiplication is a well-known routine frequently found in applications. For dense matrices with dimension n, the computation requirement of matrix multiplication is $2n^3$ and the memory requirement is $3n^2$. Thus,

$$W = 2n^3 \text{ and } M = 3n^2$$

Writing W as a function of M, we have

$$W = (\frac{2M}{3})^{3/2}$$

This means that

$$W = h(M) = (\frac{2M}{3})^{3/2}$$

Therefore,

$$W' = h(N \times M) = (\frac{2NM}{3})^{3/2} = N^{3/2} \times h(M)$$

That is,

$$g\,(N) = h(N \times M)/h\,(M) = N^{3/2}$$

In a similar manner, given the computation complexity and memory complexity, we can get the $g(N)$ value for any application. $g(N)$ represents the data reuse rate when memory is scaled N times. Table I shows the values of some applications. As will be shown later, $g(N)$ significantly impacts the optimal CMP configuration for different phases of an application and diverse applications. The merit of this work is that it presents an Analytical plus Simulation method to automatically obtain the quantitative solution before detailed simulation.

TABLE I
THE $g(N)$ FACTORS OF SOME APPLICATIONS

Application	Computation	Memory	g(N)
TMM(Tiled matrix multiplication)	N^3	N^2	$N^{3/2}$
Band sparse matrix multiplication	N	N	N
Stencil	N	N	N
FFT(Fast Fourier Transform)	N	$Nlog_2N$	$2N$

III. THE C^2-BOUND CMP DSE MODEL

We formalize the many-core design space exploration as an optimization problem. Note that object function and constraints are needed for an optimization problem. In this section, we firstly present the execution time object function and constraints for optimization. Then, we propose the methodology for automatic collection of the needed parameter values and then resolve the optimization problem. Lastly, we will discuss how the analytical results can facilitate simulation.

Given a fixed problem size, the impact of memory level concurrency and process level concurrency can be illustrated in Fig. 2, where the x-axis is time, and the y-axis is the amount of work being done in parallel. Subgraph (a) shows the case when there is only one process (p=1) and no memory concurrency (C=1). (b) shows the case when multiple processes are available (p=N) but still without memory concurrency (C=1). (c) shows the highest concurrent case when multiple processes are available (p=N) with memory concurrency (C>1). The shadowed area is the total amount of operations done. The sum of the length of all the shadowed rectangles is the time it would take to run.

Quantifying the combined effect of the memory level concurrency and process level concurrency, as demonstrated in Fig. 2, is difficult due to the entangled interaction between data access patterns and the underlying computing system. What makes the problem become even more challenging is that, the problem size is usually a function of the available memory capacity.

In this study, the bound impact of memory concurrency and memory capacity on achievable many-core performance are examined. The proposed model is called C^2-Bound, where C^2 denotes the consideration of both data access Concurrency and "memory" Capacity. Note that the memory capacity here is on-chip memory capacity (more discussion will be presented in Section V).

A. Execution Time Object Function

Let problem size (in terms of the dynamic Instruction Count) be IC. Eq. (5) is the classic formulation of CPU-time of sequential processing in terms of data stall time [21].

$$\begin{aligned} &CPU\text{-}time \\ &= IC \times (CPI_{exe} + data\text{-}stall\text{-}time) \times Cycle\text{-}time \end{aligned} \quad (5)$$

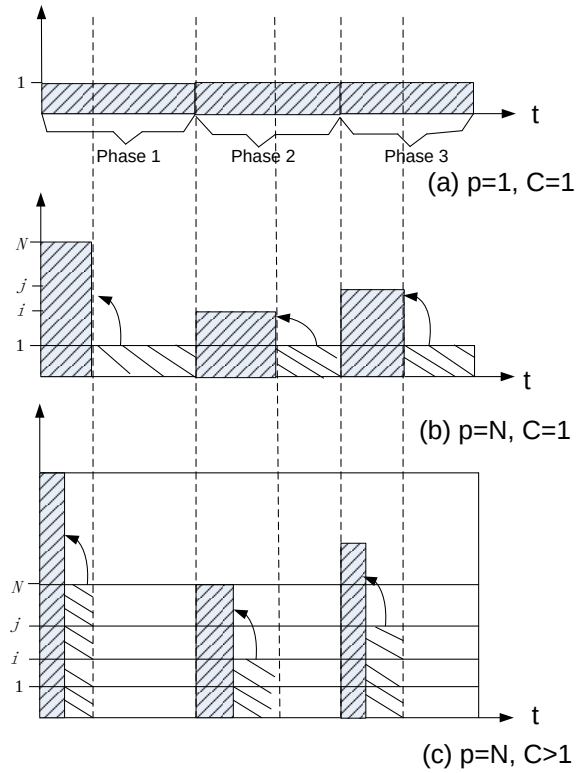

Fig. 2. The impact of process level concurrency and memory level concurrency on program running time

Eq. (6) is the conventional data stall time formula based on AMAT [21].

$$Data\text{-}stall\text{-}time = f_{mem} \times AMAT \tag{6}$$

The AMAT based Eq. (6) only considers memory locality, but not concurrency. Eq. (6) no longer holds when data access concurrency exists.

Recently, we have extended Eq. (6) to consider both locality and concurrency [20]. The extended C-AMAT based execution time is Eq. (7). A rigorous proof has been made in [20] with regard to the correctness and generality of Eq. (7) for a single processor. However, two questions remain for our C^2-Bound model: 1) extension to multiprocesors, 2) the consideration of memory capacity.

$$T = IC \times (CPI_{exe} + f_{mem} \times C\text{-}AMAT \times (1 - overlapRatio_{c\text{-}m})) \times Cycle\text{-}time \tag{7}$$

We take Eq. (7) as the start point to study the scalability issues. According to Sun-Ni's law, the execution time object function can be formed as Eq. (8).

$$J_D = T_1 + \frac{g(N) \times T_N}{N} \tag{8}$$

Where T_1 is the execution time of the serial part of the workload IC_1. T_N is the sequential execution time of parallel part of the workload IC_2. The portion of IC_1 to IC is f_{seq} and the portion of IC_2 to IC is $1 - f_{seq}$. That is,

$$IC_1 = IC \times f_{seq} \text{ and } IC_2 = IC \times (1 - f_{seq})$$

As the parallel degree i can be from 1 to N, Eq. (8) can be generalized as follows. For the brevity of discussion, we use the simple version as Eq. (8), but in real CMP DSE we have implemented the generalized version.

$$J_D = \sum_{i=1}^{N} (g(i) \times T_i / i)$$

According to Sun-Ni's Law, the problem size IC can be scalable with memory capacity, and recall that the memory capacity is increasing linearly with N, so the following relation holds, where IC_0 is the problem size when $N = 1$.

$$IC = g(N) \times IC_0 \tag{9}$$

Therefore, combining Eq. (7), (8) and (9), we can derive Eq. (10) as the object function for application execution time.

$$J_D = IC_0 \times (CPI_{exe} + f_{mem} \times C\text{-}AMAT \times (1 - overlapRatio_{c\text{-}m}))(f_{seq} + \frac{g(N) \times (1 - f_{seq})}{N}) \tag{10}$$

Eq. (10) will be used as the object function for optimization. In Eq. (10), the features of data access patterns have been denoted by C-AMAT, especially data access concurrency within a single core. Now we move onto developing the constraints for the optimization problem.

B. Physical Constraints

Fig. 3 shows a schematic illustration of CMP architecture. There are three basic components: the NoC-connected cores, the fixed function logic (timing, test, and debug), memory controllers and I/O interfaces. The cores each access their own subset of a coherent or non-coherent L2 cache to provide high-bandwidth L2 cache access.

Pollack's rule states that microprocessor performance increase due to microarchitecture advances is roughly proportional to the square root of the increase in complexity [1].

$$CPI_{exe} \propto {A_0}^{-1/2}$$

We use the rule to model the computing performance of a processor core as Eq. (11).

$$CPI_{exe} = k_0 {A_0}^{-1/2} + \phi_0 \tag{11}$$

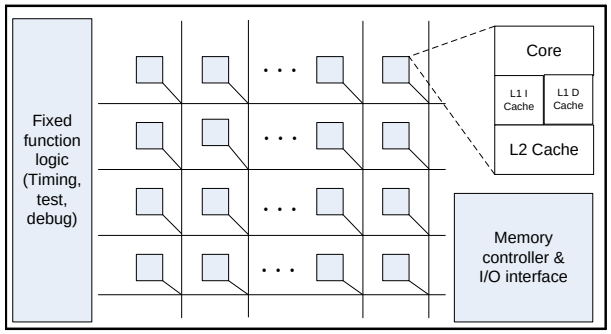

Fig. 3. Chip multiple processors

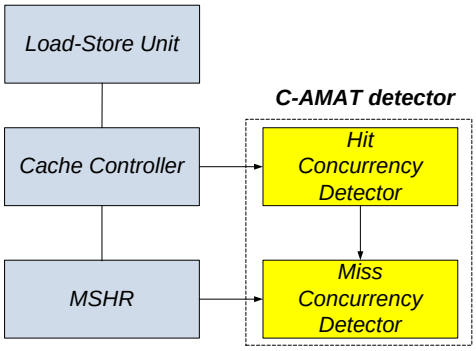

Fig. 4. C-AMAT detector

Assume the chip has A area in total. For the brevity of discussion, we also assume the processor core is symmetric. As shown in Eq. (12), all the cores have equal private areas. The case for asymmetric and dynamic multicore processors can be derived similarly.

$$A = N(A_0 + A_1 + A_2) + A_c \tag{12}$$

Where N is the number of cores in CMP, A_0 is the area of a processors core (excluding its private cache), and A_1 is the area of the private cache of a processor. A_2 is the area of the L2 cache allocated for a given processor. A_c is the area allocated for the shared functions including shared caches, interconnections, memory controllers, test and debug, etc. The object function and constraints considering both C-AMAT and Sun-Ni's Law have been discussed, now we are ready to solve the optimization problem.

C. Optimization Problem and Solving

Given that the total silicon area of the chip is fixed, the allocation of the silicon for core logic, L1 cache, and L2 cache, will influence application performance. The law of diminishing marginal utility should be considered into the allocation. Based on the object function and physical constraints, we formalize the CMP DSE optimization problem as follows.

$$\begin{cases} Min\ Eq.\ (10) \\ Sat.\ Eq.\ (12) \end{cases}$$

We solve the optimization problem using the method of Lagrange multipliers, minimizing:

$$L(A_1, A_2, \lambda, N) = J_D + \lambda[N(A_0 + A_1 + A_2) + A_c - A] \tag{13}$$

Differentiating Eq. (13) with respect to A_0, A_1, A_2, λ, N, we can build a set of nonlinear equations. When we use tools analyzing Eq. (13), we find that

$$\frac{\partial L}{\partial N} > 0 \text{ if and only if } g(N) \geq O(N)$$

Therefore the optimization falls into two cases. When $g(N) \geq O(N)$, there exists no N value to obtain the optimal execution time. In this case, we find the optimal A_0, A_1, A_2, N that maximize W/T which is the ratio of problem size and execution time. On the other hand, when $g(N) < O(N)$, we can find the optimal core number N to minimize execution time T, and its corresponding A_0, A_1, and A_2. Eq. (13) provides a good theoretical analytic result for CMP design. But, in engineering design we often need a more accurate design layout based on simulation. In the following, we discuss how the C^2-Bound model can guide CMP simulation.

D. APS Methodology

We propose the automatic APS (Analysis plus Simulation) method for C^2-Bound based CMP DSE. The APS method follows a "characterization + optimization + simulation" flow. The characterization collects the input parameters for the optimization. The input information can be obtained directly from an application development manual, analyzed by a compiler, or profiled by hardware detection structures.

Fig. 4 illustrates the C-AMAT analyzer which is a hardware detection system. The Hit Concurrency Detector (HCD) counts the total hit cycles and records each hit phase in order to calculate the average hit concurrency. The HCD also notifies the Miss Concurrency Detector (MCD) whether a current cycle has a hit access. Therefore, with the hit information from HCD and the miss information from miss status holding registers (MSHR), MCD is able to obtain the total number of pure miss cycles.

We have successfully collected all the needed input parameters for an application running on physical machines using PAPI [22] and HPCToolkit [23]. Moreover, we also achieved the same goal with the help of GEM5 [19] and DRAMSim2 [24].

Taking the parameters as inputs, Fig. 5 shows the optimization flow. The left column lists the four steps: the input, formalization, solving and the output, while the right column presents the implementation details. Note that the solution of the nonlinear equations can be found using Newton's method. We have implemented an efficient solver for the nonlinear equation set.

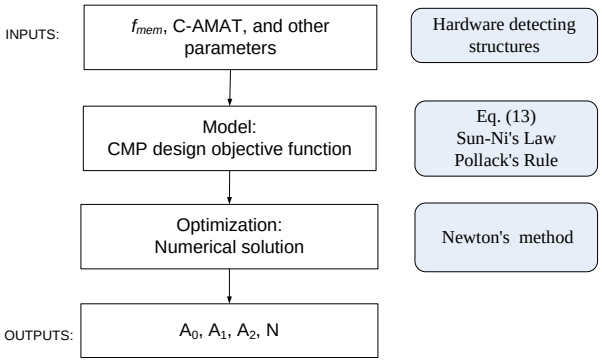

Fig. 5. Analytical methodology overview

APS Algorithm for CMP DSE

```
// Initially measure the parameters through application characterization
  For each application, using tools to measure
    f_mem, C-AMAT, and other parameters
// Building optimization equation set
Differentiating Eq. (13) with respect to A0, A1, A2, λ, N:
  // case I when the workload size is linear or super-linear scalable
  If (g(N) ≥ O(N)) Do
    Optimizing to maximize W/T
    Output A0, A1, A2, N
  // case II when the workload size is fixed or inferior-linear
  If (g(N) < O(N)) Do
    Optimizing to minimize T
    Output A0, A1, A2, N
// Simulation to refine the solution
  Simulating the adjacent regions in the design space nearby the solution presented
    by the analytical model
  Output the final solution of all the microarchitecture parameters
End
```

Fig. 6. Analysis plus simulation (APS) algorithm

The solver can be integrated with a simulator to guide detailed evaluation. Only the adjacent regions in the design space near the solution presented by the C^2-Bound model are worth the time-consuming simulation. We formally present the Analysis Plus Simulation (APS) algorithm in Fig. 6.

APS is the collaboration of analytical modeling and detailed simulation. The optimal core count, the space allocation between processing and caches, are determined by the optimization model. Once these fundamental parameters are fixed, the skeleton of CMP becomes clear. Based on the skeleton, microarchitecture parameters such as issue width and ROB size can be efficiently evaluated via simulation since the design space has been narrowed significantly.

In next section, the detailed results of the verification and case studies are presented.

IV. Validation And Case Study

The state-of-the-art cycle-accurate simulator GEM5 [19] and DRAMSim2 [24] are integrated to provide an appropriate memory performance simulation. We model a detailed 4-way out-of-order processor with a 128-entry reorder buffer, a two-level cache hierarchy. The memory hierarchy is similar to an Intel Core i7 system [25]. We also implemented on-line detecting structure for the C-AMAT analyzer shown in Fig. 4.

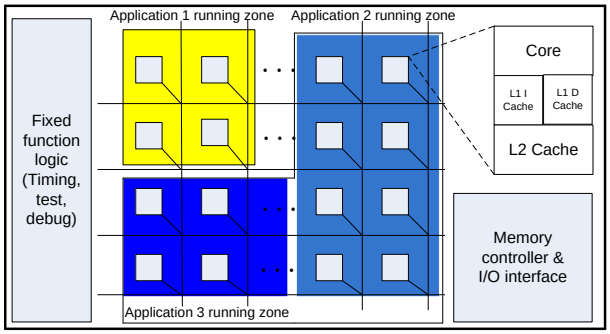

Fig. 7. Core allocation for multiple tasks in a CMP

We used benchmark suites, SPLASH-2 and PARSEC, which have several input datasets at different scales [12] [13]. Aided by SimPoint [26], 10 billion dynamic instructions for each benchmark were simulated to collect statistics.

Recent product announcements show a trend toward aggressive integration of many cores on a single chip to maximize throughput. However, efficiently utilizing many resources is not easy. The behavior of an application changes phase by phase during its execution. There is no fixed hardware configuration that can work best for all the possible behaviors. Each design has its own pros and cons, depending on the interaction between data access patterns and the underlying memory system.

Fortunately, programs have periodic behaviors and their data access patterns are predictable [26]. With a set of lightweight counters, we are able to deploy proper optimization techniques to timely adapt to the underlying data access pattern changes of an application.

C^2-Bound analytic results can be either used in reconfigurable hardware environments or, by software designers, applied to scheduling, partitioning, and allocating resources among diverse applications. Fig. 7 includes three applications. As the sequential portion f_{seq} is very large and memory concurrency C is very low, the first application needs the least number of cores and thus the benefit for allocating more cores to the first application is marginal. On the other hand, the second application has a low f_{seq} and a high C. Therefore, it is sensible to assign more cores to the second application. The third application falls somewhere between these two extremes. In this manner, the application demand can be well matched into the underlying hardware.

In the following section, we will discuss the optimal core numbers when workload is super linearly scalable, that is $g(N) = N^{3/2}$ which is reprsentative for a large number of applications. We will discuss three levels of memory concurrency: one has no memory concurrency, i.e. $C = 1$; one has a moderate memory concurrency, $C = 4$; the last is high memory concurrency $C = 8$.

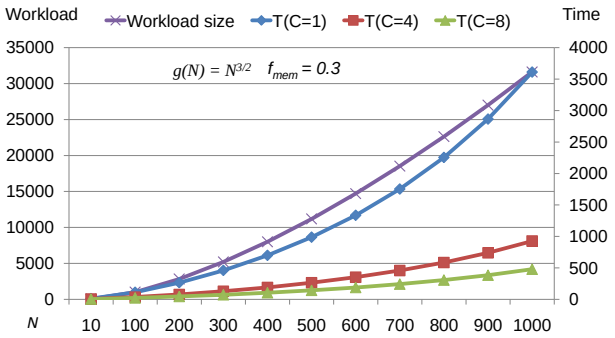

Fig. 8. The problem size W and execution time T of memory bounded scaling ($g(N) = N^{3/2}$, f_{mem}=0.3)

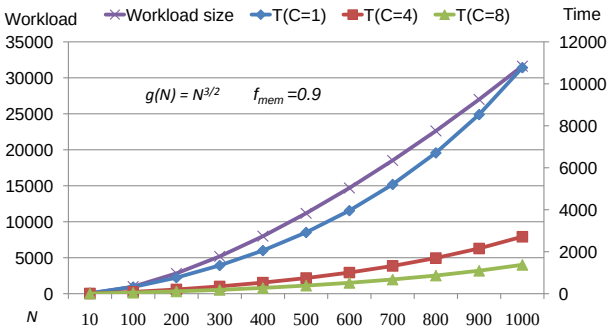

Fig. 9. The problem size W and execution time T of memory bounded scaling ($g(N) = N^{3/2}$, f_{mem}=0.9)

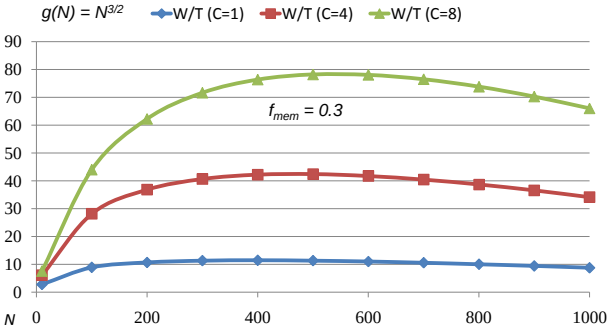

Fig. 10. The W/T of fixed problem size scaling ($g(N) = N^{3/2}$, f_{mem}=0.3)

The purpose in this section is not to present all the results of the model, but only to verify its correctness and effectiveness. We have implemented the model online to account for changes in applications' data access patterns.

Fig. 8 and 9 show the problem size W and the execution time T of memory bounded scaling when $g(N) = N^{3/2}$ and data access frequency f_{mem} is 0.3 or 0.9. Comparing Fig. 8 and 9, we find that the execution time (T) increases with data access frequency f_{mem}, Fig. 10 and 11 show the throughput (W/T) values correspondingly. Comparing Fig. 10 and 11, we find that the throughput (W/T) decreases with data access frequency f_{mem}.

When there is no memory concurrency (C=1), the scalability curve of the execution time T is close to the problem size curve. On the other hand, higher memory concurrency leads to a better scalability, in terms of execution time. For example, as shown in Fig. 8 and 9, when N is 1000, the speedup ratio of $T(C = 8)$ over $T(C = 1)$ is very significant. This tells us, even with a fixed number of processing cores, improving data access performance via memory concurrency can obtain significantly speedup. This fact is very important for the design of future supercomputer.

Fig. 10 and 11 show that when $g(N) \geq O(N)$, higher memory concurrency makes many-core computing more efficient, in terms of increasing throughput W/T. When there is no memory concurrency (C=1), about one hundred cores are enough to achieve the best throughput. When N is more than 100, the ratio of W and T remains approximately the same. However, when memory concurrency increases, W/T increases and fluctuates, so we can find an optimal point to achieve the best throughput to foster the utilization of many-core processors.

The results presented in Fig. 8 to 11 demonstrate the importance of memory concurrency and its relation with the number of cores. In general, more cores correspond to smaller cache area; the results for area allocation for cache are not presented here due to the page limitation, but can be obtained at the same time with the optimization of the number of cores.

Therefore, the optimal core count N, the space allocation between processing and caches, A_1, A_2, have been determined by our optimization model, Eq. (13). Since these are the most fundamental parameters of CMP, the skeleton of CMP becomes clear. Based on the skeleton, microarchitecture parameters such as issue width and reorder buffer (ROB) size can be efficiently evaluated via detailed simulation since the design space has been narrowed significantly.

Using the cycle-accurate simulator GEM5 [19], we have done a DSE to find the optimal chip configurations for the fluidanimate benchmark from PARSEC [13]. The fluidanimate is a computer animation application with large working sets. Six parameters (A_0, A_1, A_2, N, issue width, ROB size) are considered and each parameter has ten optional values, so the whole design space size is one million (10^6). As shown in Fig. 12, with the help of the C^2-Bound analysis, we do not need to run simulations to explore the A_0, A_1, A_2, N

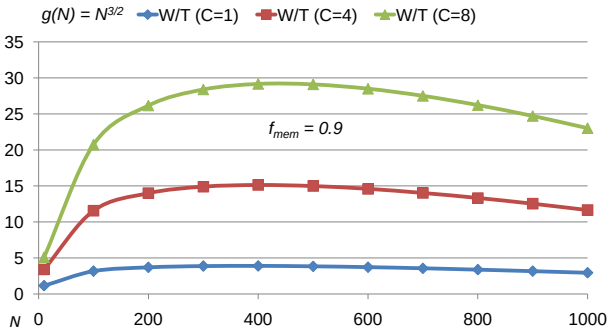

Fig. 11. The W/T of fixed problem size scaling ($g(N) = N^{3/2}$, f_{mem}=0.9)

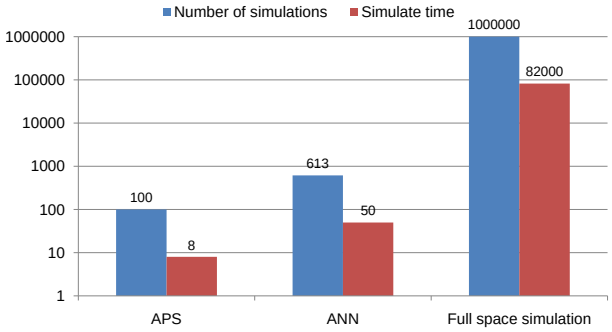

Fig. 12. The number of simulation times

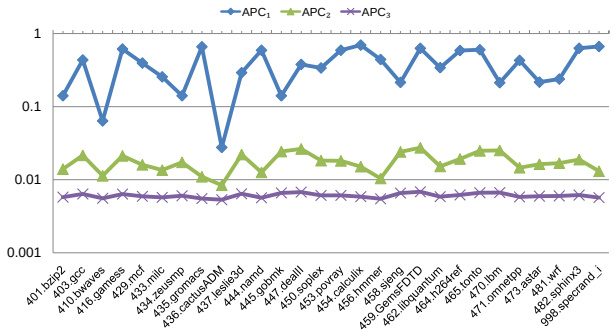

Fig. 13. The APC values at each layer of a memory hierarchy

parameters. For the rest parameters, issue width and ROB size, only one hundred (10^2) simulations are needed. Therefore, the design space has been narrowed significantly by up to four orders of magnitude, from one million to one hundred. To evaluate the accuracy of the APS method, we run simulations to traverse the full design space, which use 128 Intel Xeon processors running for 4 weeks. Then we get performance data for each of the 10^6 different configurations, with which the APS performance data are compared, and the error is 5.96%. The error may come from Pollacks rule in Eq. (11) which is an empirical equation rather a law. Note APS only used about one hour with 8 processors rather than 4 weeks with 128 processors. Compared to the time saving, the 5.96% error is acceptable.

We also use the well-known machine learning method ANN [2] to predict the performance data in the huge design space. To achieve the same prediction accuracy (5.96% error), ANN needs 613 times of simulation. Therefore, APS used only 16.3% of the simulation time to achieve the same prediction accuracy as ANN.

V. Discussion

For a many-core processor, a fundamental question is which layer of a memory hierarchy is the primary performance bound vital for many-core performance. To answer this question, we should consider three factors, latency, bandwidth and capacity simultaneously.

Initially, the interactions among the three factors are not straightforward. Fortunately, the APC metric can be used to represent the combined impact of latency and bandwidth [27]. More interestingly, the previously used metric C-AMAT = 1/APC [15].

APC (data Access Per memory-active Cycle) is a new metric to measure memory systems performance, which can be applied on each memory layer and considers both memory locality and concurrency [27]. In Fig. 13, the APC_1 is the APC value of the L1 cache, APC_2 is that of LLC (last level cache), and APC_3 is that of main memory. As the big gap between the performance of on and off-chip cache has been shown in Fig. 13, it is reasonable to conclude that in our C^2-Bound model the memory bound is the on chip memory bound. The "on-chip memory" is LLC for inclusive caches, or the sum of all the on-chip caches for exclusive caches.

Assuming the on-chip memory size is X, the working set size [28] is Y, and the problem size is Z, we can obtain the maximum value of the LLC bounded problem size via solving the algorithm shown as follows.

$$\begin{cases} Max\ Z \\ Sat.\ Y \leq X \end{cases}$$

The rationale behind the algorithm is to fit the working set into on-chip memory, we should keep the working set size no more than the on-chip cache size; otherwise the chip performance will be decreased significantly [28] [29]. Assume the on-chip memory bounded problem size is a and the real problem size is b. There exist two cases:

1) If b is no more than a, the application performance will be processor-bound. The working set of the application is captured on chip, and thus the application requires few off-chip accesses, and then the application performance is largely insensitive to on-chip memory capacity and concurrency.

2) If b is more than a, the application performance will be limited strictly by the rate that data can be moved between the processor and the DRAM. Now, cache capacity and data access concurrency will impact the application performance more significantly. A big data application is deemed a large working set application, and is likely falling into this case.

Applications may move between these two cases phase by phase, since their data access behavior may be dynamic. Therefore, reconfigurable hardware or management software (for scheduling, partitioning and allocating) is called for to achieve the dynamic matching between application and underlying hardware. Fortunately, an associated methodology in Fig. 4 has been given to obtain the needed parameters online to facilitate the C^2-Bound model being used for these purposes.

Our model considers two new parameters: data access concurrency and memory capacity-bounded problem size. The introduction of data access concurrency is based on the fact that concurrent data access exists at each layer of a memory hierarchy, and its impact should be considered in many-core design. The parameter memory capacity-bounded problem size is a vital factor in the scalability study, where increasing the number of cores is a common choice of many-core design. The inclusion of these two new parameters makes the newly

proposed C^2-Bound model significantly more appropriate for DSE than the existing locality-only and/or problem size fixed models for modern application-specific many-core design.

VI. Related Work

The many-core design exploration is a process to find an architecture with features that can well match application characteristic, and then to utilize chip space efficiently and to achieve excellent performance. Simulation and analytical modeling are the two basic approaches to accomplish this purpose. However, the simulation is costly and slow, typically a one minute execution of a real machine requires approximately one month to a year to simulate [30], an order of 10^5 to 10^6 increase in execution time. The huge design space and the high simulation cost prevent computer architects from exploring the intractable design space thoroughly. The conventional brute force simulate-compare design process becomes painfully slow, if not infeasible, to find an optimal multi-core architecture.

While simulation is an important step toward implementation, analytical methods can rapidly narrow the system design space prior to the detailed simulation. They illuminate high-level design trade-offs and present solutions for optimal performance and efficiency.

As an analytical approach, the C^2-Bound analytical model introduced in this study is effective and new. Based on the C^2-Bound model, the newly proposed APS tool adopts an integrated analysis and simulation approach to utilize the merits of both methods. It has reduced the total simulation time significantly, in an order of four folds.

Researchers have investigated analytical methods for optimizing CMP architectures. Some methods used in the machine leaning domain have been used for analytical modeling such as genetic algorithms (GA) [31] and response surface modeling (RSM) [32]. The GA and RSM are both closed-form expressions. As a result, the impacts of problem size and memory concurrency cannot be explicitly discussed.

Some open-form expressions were proposed without considering the variations of problem size and memory concurrency. Notably, work by Hill and Marty uses a measure of processor performance to augment Amdahl's law, and applies it to evaluate symmetric, asymmetric and dynamic multi-core processors [6]. In the work, the problem size is assumed to be fixed and the impact of memory concurrency is ignored.

Sun and Ni proposed the memory-bounded parallel speedup model, which is also known as Sun-Ni's law [11]. The law revealed that the scalability of computing is bounded by problem size which is limited by memory capacity. The law which is valid for supercomputing also presents insights for CMP design. However, it needs to be revisited by taking into the account of data access delay besides the problem size, and to consider the CMP features, especially the physical resource constraints [33].

With the consideration of memory-bounded problem size, Sun and Chen discussed the Sun-Ni's law based on the same CMP cost model presented by Hill and Marty and obtained very different and more optimistic results [8]. Their results are important in locality based system design. Data access concurrency, however, was not explicitly incorporated.

Cassidy and Andreou incorporated sequential data access delay in terms of AMAT into Amdahl's law [3] [4] [5]. This work take data access patterns account into an analytical model. However, Cassidy and Andreou did not consider concurrent data accesses, and also they assumed fixed problem size. Their work can be taken as special cases of the newly proposed C^2-Bound model when there exists no memory concurrency or the problem size is fixed.

Our work is fundamentally different with the above investigations. We do not keep the assumptions that problem size is fixed and the memory access is sequential. For the first time, Amdahl's law is reevaluated with the simultaneously consideration of memory concurrency and memory capacity for the silicon area constrained many-core processor design.

VII. Conclusions

While the number of transistors in a given die increases based on Moore's law, utilizing these transistors continues to be a challenging task in VLSI design. This is especially true in recent years when data access becomes the premier performance bottleneck of computing systems. To respond to the increasing importance and complexity of modern many-core architecture and memory systems, in this study, for the first time, memory concurrency and memory-bounded problem size are incorporated into many-core processor design space exploration. While maintaining simplicity and practical feasibility, with the consideration of both memory-concurrency and memory-bound, the newly proposed C^2-Bound model is significantly more accurate and more powerful than existing DSE models. It facilitates the studies of many-core data processing, workload scalability, and therefore reshapes the on-chip area allocation for processing cores, caches and memory controllers. The C^2-Bound model has been implemented and can be executed automatically under the newly proposed analysis plus simulation (APS) algorithm for fast and accurate CMP DSE, with a combination of analysis and simulation. Analytic and implementation results show C^2-Bound is feasible and effective. The analytical results have narrowed the design space significantly by up to four orders of magnitude. APS uses only 16.3% of the simulation time to achieve the same prediction result as the widely-used standard machine learning method, ANN [2], for the fluidanimate benchmark.

The extension of CMP DSE to consider the concurrency driven data access latency and memory capacity bounded problem size is a complicated process. In this study, we have used our cumulated long time experience in memory bounded formulation and in C-AMAT development. While the C^2-Bound model is essential for next generation data-centric processor design and for Exascale system design, it is only the first step in considering data and scalability in CMP DSE. Moreover, energy consumption and temperature can be considered for multi-objective exploration in future

refined versions. The extension of C^2-Bound to asymmetric CMP DSE is straightforward.

The analytic results presented in this study also can be used in hardware reconfiguration environments or used by software designers for scheduling, partitioning and allocating resource to achieve the dynamic matching between application behaviors and underlying hardware.

In this paper, we only focus on data access concurrency and memory bounded problem size for high performance computing. In the future, more objects can be included in such an analysis. For example, the object function in Eq. (10) can be reshaped to achieve a balance among performance, power, energy and temperature [34] [35].

ACKNOWLEGMENT

This work is supported in part by the National Science Foundation, under grant CCF-1536079, grant CCF-0937877 and grant CNS-0751200.

REFERENCES

[1] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 746–749.

[2] E. İpek, S. A. McKee, R. Caruana, B. R. de Supinski, and M. Schulz, "Efficiently exploring architectural design spaces via predictive modeling," in *Proceedings of the 12th international conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2006.

[3] A. Cassidy and A. G. Andreou, "Analytical methods for the design and optimization of chip-multiprocessor architectures," in *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*. IEEE, 2009, pp. 482–487.

[4] A. Cassidy, K. Yu, H. Zhou, and A. G. Andreou, "A high-level analytical model for application specific cmp design exploration," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 2011, pp. 1–6.

[5] A. S. Cassidy and A. G. Andreou, "Beyond amdahl's law: an objective function that links multiprocessor performance gains to delay and energy," *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1110–1126, 2012.

[6] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, no. 7, pp. 33–38, 2008.

[7] D. H. Woo and H.-H. S. Lee, "Extending amdahl's law for energy-efficient computing in the many-core era," *Computer*, no. 12, pp. 24–31, 2008.

[8] X.-H. Sun and Y. Chen, "Reevaluating amdahl's law in the multicore era," *Journal of Parallel and Distributed Computing*, vol. 70, no. 2, pp. 183–188, 2010.

[9] N. Hardavellas, I. Pandis, R. Johnson, N. Mancheril, A. Ailamaki, and B. Falsafi, "Database Servers on Chip Multiprocessors: Limitations and Opportunities," in *Proceedings of the Biennial Conference on Innovative Data Systems Research*, no. 8, 2007.

[10] S. Somogyi, T. F. Wenisch, A. Ailamaki, and B. Falsafi, "Spatio-temporal Memory Streaming," in *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3. ACM, 2009, pp. 69–80.

[11] X.-H. Sun and L. M. Ni, "Another view on parallel speedup," in *Proceedings of Supercomputing'90*. IEEE, 1990, pp. 324–333.

[12] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," in *ACM SIGARCH Computer Architecture News*, vol. 23, no. 2. ACM, 1995, pp. 24–36.

[13] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. ACM, 2008, pp. 72–81.

[14] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[15] X.-H. Sun and D. Wang, "Concurrent average memory access time," *IEEE Computer*, vol. 47, no. 5, pp. 74–80, 2014.

[16] X.-H. Sun, "Concurrent-AMAT: A Mathematical Model for Big Data access," *HPC Magazine*, 2014.

[17] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM, 1967, pp. 483–485.

[18] J. L. Gustafson, "Reevaluating amdahl's law," *Communications of the ACM*, vol. 31, no. 5, pp. 532–533, 1988.

[19] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[20] Y.-H. Liu and X.-H. Sun, "Reevaluating data stall time with the consideration of data access concurrency," *Journal of Computer Science and Technology*, vol. 30, no. 2, pp. 227–245, 2015.

[21] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2012.

[22] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A portable programming interface for performance evaluation on modern processors," *International Journal of High Performance Computing Applications*, vol. 14, no. 3, pp. 189–204, 2000.

[23] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent, "Hpctoolkit: Tools for performance analysis of optimized parallel programs," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, pp. 685–701, 2010.

[24] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "Dramsim2: A cycle accurate memory system simulator," *Computer Architecture Letters*, vol. 10, no. 1, pp. 16–19, 2011.

[25] D. Levinthal, "Performance analysis guide for intel core i7 processor and intel xeon 5500 processors," *Intel Performance Analysis Guide*, 2009.

[26] G. Hamerly, E. Perelman, and B. Calder, "How to use simpoint to pick simulation points," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 4, pp. 25–30, 2004.

[27] D. Wang and X. Sun, "Apc: A novel memory metric and measurement methodology for modern memory system," *IEEE Transactions on Computers*, vol. 63, no. 7, pp. 1626–1639, 2014.

[28] P. J. Denning, "The working set model for program behavior," *Communications of the ACM*, vol. 11, no. 5, pp. 323–333, 1968.

[29] ——, "The locality principle," *Communications of the ACM*, vol. 48, no. 7, pp. 19–24, 2005.

[30] L. Eeckhout, "Computer architecture performance evaluation methods," *Synthesis Lectures on Computer Architecture*, vol. 5, no. 1, pp. 1–145, 2010.

[31] M. Thompson and A. D. Pimentel, "Exploiting domain knowledge in system-level mpsoc design space exploration," *Journal of Systems Architecture*, vol. 59, no. 7, pp. 351–360, 2013.

[32] G. Palermo, C. Silvano, and V. Zaccaria, "Respir: a response surface-based pareto iterative refinement for application-specific design space exploration," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 12, pp. 1816–1829, 2009.

[33] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron, "Cmp design space exploration subject to physical constraints," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.* IEEE, 2006, pp. 17–28.

[34] S. Cho and R. G. Melhem, "Corollaries to amdahl's law for energy," *Computer Architecture Letters*, vol. 7, no. 1, pp. 25–28, 2008.

[35] W. Huang, K. Skadron, S. Gurumurthi, R. J. Ribando, and M. R. Stan, "Exploring the thermal impact on manycore processor performance," in *Semiconductor Thermal Measurement and Management Symposium, 2010. SEMI-THERM 2010. 26th Annual IEEE*. IEEE, 2010, pp. 191–197.