Moguls: a Model to Explore the Memory Hierarchy for Bandwidth Improvements

Guangyu Sun[†], Christopher Hughes[§], Changkyu Kim[§],
Jishen Zhao[†], Cong Xu[†], Yuan Xie[†], Yen-Kuang Chen[§]
[†]Pennsylvania State University[*] and [§]Intel Labs
{gsun, juz138, czx102, yuanxie}@cse.psu.edu
{christopher.j.hughes, changkyu.kim, yen-kuang.chen}@intel.com

ABSTRACT

In recent years, the increasing number of processor cores and limited increases in main memory bandwidth have led to the problem of the *bandwidth wall*, where memory bandwidth is becoming a performance bottleneck. This is especially true for emerging latency-insensitive, bandwidth-sensitive applications. Designing the memory hierarchy for a platform with an emphasis on maximizing bandwidth within a fixed power budget becomes one of the key challenges. To facilitate architects to quickly explore the design space of memory hierarchies, we propose an analytical performance model called Moguls. The Moguls model estimates the performance of an application on a system, using the bandwidth demand of the application for a range of cache capacities and the bandwidth provided by the system with those capacities. We show how to extend this model with appropriate approximations to optimize a cache hierarchy under a power constraint. The results show how many levels of cache should be designed, and what the capacity, bandwidth, and technology of each level should be. In addition, we study memory hierarchy design with hybrid memory technologies, which shows the benefits of using multiple technologies for future computing systems.

Categories and Subject Descriptors

B.3.3 [**Memory Structures**]: Performance Analysis and Design Aids—*Formal models, Simulation*

General Terms

Design, Performance

Keywords

Memory model, bandwidth, memory hierarchy, throughput computing, power consumption

[*]G. Sun, J. Zhao, C. Xu, and Y. Xie are supported in part by NSF CCF-0903432, CNS-0905365, SRC grant, and DoE's ASCR program under award number DE-SC0005026.

ISCA'11, June 4–8, 2011, San Jose, California, USA.

1. INTRODUCTION

Recently, the importance of applications that are bandwidth-sensitive and latency-insensitive has increased significantly. At the same time, the growth rate of the computation abilities of a processor has continued to outstrip the growth rate of main memory bandwidth. This has led to a *bandwidth wall*.

A key subset of bandwidth-sensitive and latency-insensitive applications is known as throughput computing applications. Throughput computing refers to trading off single thread performance for higher overall computational throughput. Throughput computing applications span many domains and are already critical on a variety of platforms, including high-performance computing (HPC) machines (e.g., molecular dynamics simulations [1]), commercial servers (e.g., database query processing [2]), and client machines (e.g., image/video processing [3]). Memory bandwidth is critical to many throughput computing applications because of two main reasons:

- Throughput computing applications inherently have generous amounts of parallelism that processors can take advantage of via multi-threading and single-instruction-multiple-data (SIMD) execution; thus, hardware can consume data at high rates. Systems designed to perform well on throughput computing applications achieve high performance by exploiting their inherent parallelism. These systems support large numbers of threads and/or use wide SIMD execution (e.g., Sun's Niagara [4] and nVidia's Tesla [5]), which puts a lot of pressure on the memory system. In throughput computing, memory latency is typically not a bottleneck since the latency can be hidden via multithreading (e.g., for GPUs) or hardware prefetching (e.g., for CPUs). However, bandwidth is a potential bottleneck.
- Many throughput computing applications have inherently large working sets (e.g., tens to hundreds of MB), which are unlikely to fit in conventional on-die SRAM caches for the foreseeable future. Further, unlike more traditional workloads (e.g., those similar to TPC benchmarks), some throughput computing applications show a sharp drop in performance once caches are too small to hold their working sets - we refer to this as a performance cliff. Thus, these applications are likely to be bandwidth-bound at main memory unless some significant changes are made to the memory hierarchy. Even in the general-purpose computing community, memory bandwidth was predicted to become a performance bottleneck [6], for example, due to reduced bandwidth efficiency from overly-aggressive hardware prefetching.

Throughput computing can be conducted on either CPU-based or GPU-based systems. CPUs historically have used multi-level cache hierarchies to reduce average latency and reduce bandwidth demands on larger capacity levels. GPUs have so far relied on shallow hierarchies—they hide latency with multithreading and use much higher bandwidth in main memory than CPUs (i.e., GDDR vs. DDR). The common wisdom is that the GPU approach is better for throughput computing applications. While this may be true, GPUs are designed primarily for graphics. Consequently, neither GPUs nor CPUs have a memory hierarchy designed specifically for throughput computing with an emphasis on bandwidth improvement. There are a few simple techniques to improve bandwidth efficiency of a system [7], but they are insufficient for the large bandwidth requirements of some throughput computing applications. We also cannot rely on ever-increasing main memory bandwidth to meet the needs of throughput computing. For example, today's systems that tout good performance for throughput computing applications (e.g., nVidia's Tesla) do so by providing large main memory bandwidth via the use of GDDR rather than improving bandwidth efficiency. However, GDDR has fairly strict capacity limits and is much more power hungry than conventional DRAM modules, which reduces its desirability for throughput computing, and makes it an unfavorable choice for general-purpose systems trying to improve their throughput computing performance. Furthermore, technology trends indicate that growth in bandwidth demand outpaces growth in bandwidth supply for all DRAM-based memory. (For example, historically processor throughput has grown by a factor of 1.5x per year, while DRAM bandwidth has grown by only 1.3x per year [8]). Consequently, it is necessary to study the memory hierarchy design for throughput computing platforms with an emphasis on bandwidth improvement.

Our Contribution. Architects would like to optimize the memory hierarchy design with bandwidth improvements as the first priority design goal and try to find out (1) the number of levels in the optimal cache hierarchy, (2) the capacity and bandwidth of each level, and (3) the appropriate memory technology (SRAM, eDRAM, or other emerging memory technologies) of each level. To help computer architects quickly explore the design space for memory hierarchies to improve bandwidth, this paper makes the following contributions.

- *An analytical model called Moguls is proposed to quickly estimate the performance for applications on specific memory hierarchy designs.* The model is based on the *bandwidth demand* of an application and the *bandwidth provided* by the memory hierarchy design. The bandwidth demand/provided is defined at all memory capacities, and is described as a *capacity-bandwidth (CB)* curve (Section 2). The CB curve can facilitate a quick estimation of whether the bandwidth provided by the memory hierarchy can satisfy the bandwidth demand, and guide design improvements (for example, estimating the impact of changing the capacity of existing levels of memory, or of adding extra levels of memory).
- *The usefulness and the effectiveness of the model is demonstrated by exploring the memory hierarchy design for multi-programmed workloads running on a high throughput processor.* First-order approximations are proposed to help us quantitatively determine: (a) the most energy-efficient capacity and bandwidth of the levels in the cache hierarchy, (b) the optimal number of cache levels, (c) potential performance benefits of multiple levels of memory assuming a fixed power budget. That is, given a power budget, our model and theories suggest the best memory hierarchy (in terms of capacity and bandwidth provided). We also explore how to choose the optimal levels, capacity, and bandwidth when multiple memory technologies are provided. In order to validate our theories, we find the optimal memory hierarchy design with exhaustive simulations for real cases. We show that our theories match experimental results.

2. MOGULS MEMORY MODEL

Designing a memory hierarchy to maximize performance and minimize power consumption is very challenging. How many levels of cache should we have? What capacities should they have? What bandwidths should they provide? On which memory technology should they based? To help solve this complex optimization problem, we first propose a way to analytically model a memory system's specifications and an application's memory requirements. We later show how to use this model to design an optimal memory hierarchy.

2.1 Problem Description

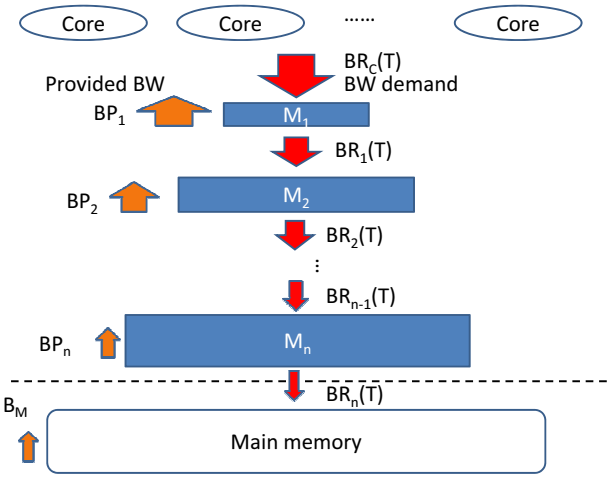

Figure 1: Bandwidth requirements of the memory hierarchy.

Figure 1 shows a computing system with multiple cores and a multi-level memory hierarchy. The hierarchy has n levels of shared cache (M_1 to M_n). Each level of cache can provide bandwidth BP_i to the next higher level of cache. Main memory provides bandwidth B_M. The system runs an application with a peak instruction throughput of T. This results in an aggregate bandwidth requirement from the cores, BR_C. Each level of cache filters out some of the requests, reducing the bandwidth requirement out of level i to (BR_i). In order to actually achieve a throughput of T, the bandwidth requirement of each level must be met by the level below:

$$BR_C(T) \leq BP_1; \quad BR_i(T) \leq BP_{i+1} \ (1 \leq i < n); \quad BR_n(T) \leq B_M \qquad (1)$$

If the bandwidth requirement out of a level is greater than the bandwidth provided by the level below, the application is bandwidth-bound and the achieved throughput (i.e., performance) is below the system's peak.

Similar reasoning applies for systems with private caches or caches shared by only a subset of cores—the bandwidth requirement of each core (or subset of cores) must be separately satisfied to see peak throughput. Consequently, we discuss only shared caches, but our conclusions can be extended to cover other types of cache hierarchies.

2.2 Moguls Memory Model

We now use the concepts from Figure 2 to construct a model, called *Moguls*, to reason about memory systems considering the

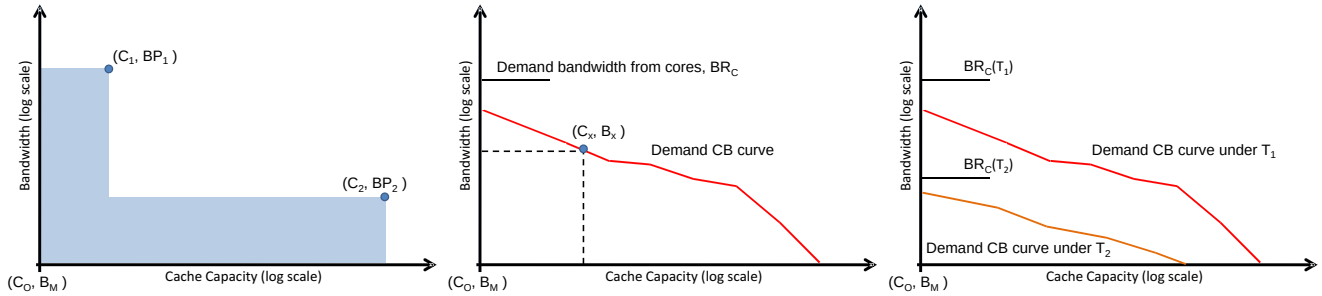

Figure 2: CB curves from the Moguls model: (a) the provided CB curve of a system; (b) the demand CB curve of an application; (c) two demand CB curves of the same application at different computing throughputs.

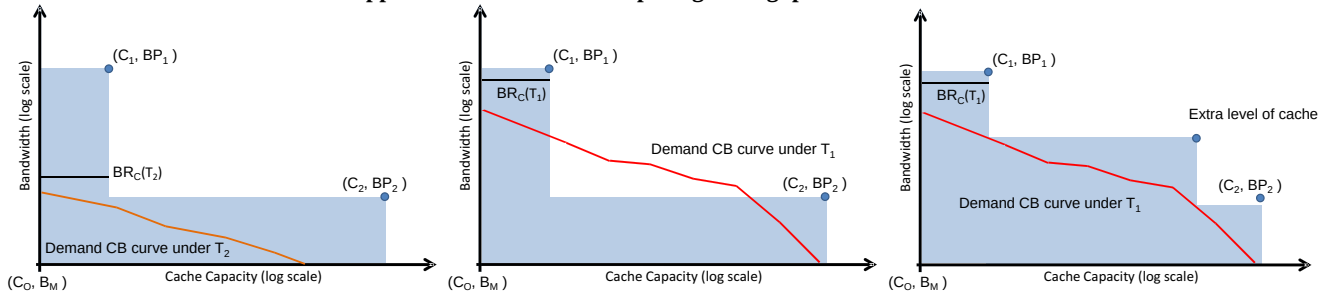

Figure 3: (a) A demand CB curve is satisfied by a provided CB curve; (b) a demand CB curve is NOT satisfied by a provided CB curve; (c) a demand CB curve is satisfied by a provided CB curve by adding an extra level of cache.

bandwidth requirements. The Moguls model maps a computing system and an application's characteristics to capacity-bandwidth (CB) coordinates, as shown in Figure 2. For each point in CB-space, the *y*-axis represents the required/provided bandwidth of a cache level, and the *x*-axis represents the capacity of the cache level. In the Moguls model, two *CB curves* are used to estimate the throughput of an application running on a system: (1) the system's provided CB curve, (2) the application's demand CB curve.

The *provided CB curve* is defined by the capacity and the effective bandwidth provided by each level of a system's memory hierarchy, and typically forms a "stair-case" shape. The bandwidth a system provides at a given capacity is determined by the level of the memory hierarchy with that capacity, or the next largest capacity if no level exists with that capacity. Figure 2(a) shows an example of a provided CB curve (the boundary of the shaded portion) for a system with two levels of cache—one is relatively small but with high bandwidth, and the other is relatively large but with low bandwidth. Note that the origin of the coordinate system is the point (C_O, B_M), where C_O represents the minimum capacity of a cache design that is available, and B_M is the bandwidth provided by main memory.

The *demand CB curve* represents an application's capacity and bandwidth requirements—if those requirements are met, the application sees the same performance it would on a memory system with infinite capacity and bandwidth. The demand curve is defined by an application's working set sizes and the rate of execution of loads and stores. It can be derived from what is commonly referred to as a working set plot, which shows cache miss rate vs. cache capacity. The working set plot is then scaled to factor in the rate that the application's loads and stores are executed in the underlying hardware (assuming infinite bandwidth provided). Figure 2(b) gives an example demand CB curve. The point (C_x, B_x) on the curve represents: *given a cache with capacity C_x, the required bandwidth to the next level down in the memory hierarchy is B_x*. The bandwidth requirement of the cores, $BR_C(T)$, is shown in the figure on the y axis. Note that since the scale is log-log, the y-axis is *not* at a cache capacity of zero; thus, BR_C is slightly above the y-intercept of the demand CB curve. $BR_C(T)$ is a function of the cores' throughput, as previously described. In Figure 2(c), two demand CB curves are shown for the same application for different computing throughputs—a larger computing throughput naturally results in a higher bandwidth requirement.

Figure 3 shows how we combine the provided and demand CB curves to determine whether the application bandwidth requirements are satisfied. In Figure 3(a), the provided CB curve is strictly above the demand CB curve—each level of the memory hierarchy can satisfy the bandwidth requirement from the level above, as defined in *Equation (1)*. In Figure 3(b), however, the provided CB curve dips below the demand CB curve. Consequently, the second level cache cannot satisfy the bandwidth requirement from the first level cache; thus, the system fails to achieve throughput T_1 with the example two-level cache design.

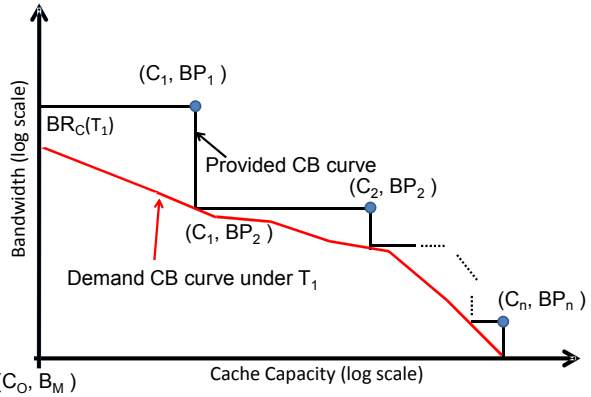

Figure 4: Generating a provided CB curve from a given demand CB curve.

In the example, we can modify the two level hierarchy in three ways to satisfy *Equation (1)*: (1) increase the bandwidth of the larger, lower bandwidth level, (2) increase the capacity of the smaller, higher bandwidth level, or (3) add one or more new levels to the hierarchy with bandwidths and capacities between the two existing levels. Figure 3(c) illustrates the latter option.

We call our model the Moguls model because if we view the provided CB curve as a ski slope, a hierarchy with many levels has many bumps (corners), known as moguls. A memory hierarchy with smaller bumps (i.e., more graceful bandwidth degradation) is likely to provide higher performance, just as a ski slope with smaller bumps allows for faster skiing.

2.3 Generation of Provided CB curve

We now describe how to generate a provided CB curve to match a given demand CB curve (i.e., how to design a cache hierarchy to just meet an application's bandwidth requirements). The cache capacities are chosen arbitrarily here; later, we will address how to choose optimal capacities.

- **Step 1** The first level cache must provide bandwidth of at least $BR_C(T)$ to satisfy the cores' requirements. As shown in Figure 4, we choose the point (C_1, BP_1) for the first level cache. The point (C_1, BP_1) represents a cache level with capacity C_1 and can provide bandwidth BP_1.
- **Step 2** After choosing the capacity and bandwidth for a level, we draw a vertical line from the point representing the cache till it hits the demand CB curve. The y coordinate of this intersection point is the bandwidth requirement of that cache; thus, this defines the bandwidth provided for the next level. For example, the point (C_2, BP_2) in Figure 4 is chosen as the next level. Note that the corner point (C_1, BP_2) is on the demand CB curve.
- **Step 3** Step 2 is repeated till the bandwidth requirement is no more than B_M. The provided CB curve is obtained by connecting these points together, as shown in Figure 4.

3. DESIGNING A MEMORY HIERARCHY WITH MOGULS

In this section, we show how to apply the Moguls model to help design a memory hierarchy optimized for power and performance. The model itself is a tool to help us analyze behavior of a specific, well-understood workload for different points in the memory system design space. To design an energy-efficient memory hierarchy for a large workload domain, we first augment the model with some assumptions about (a) the workload characteristics, and (b) power consumption of memory.

3.1 Approximations Used to Apply the Moguls Model

Approximation-1 *The demand CB curve is represented as a straight line with slope $-\frac{1}{2}$ in log-log space.*

Since it may be impractical to collect the detailed memory requirements of even one workload of interest (not to mention a whole workload domain), we approximate them. We represent target workloads with a single demand CB curve that follows the so-called 2-to-$\sqrt{2}$ rule [9]. This rule of thumb says that for complex workloads, cache miss rate varies with cache size according to an inverse power law, where the power is -0.5—if cache size is doubled, miss rate drops by a factor of $\sqrt{2}$. Studies have shown that the power actually lies between -0.3 and -0.7 [9]. Figure 5(a) illustrates the cache miss rates of several multi-programmed workloads for various cache capacities. An idealized 2-to-$\sqrt{2}$ curve (the dashed line) is also shown for comparison. Since the data is plotted on a log-log scale, the 2-to-$\sqrt{2}$ curve is a straight line with slope $-\frac{1}{2}$. This line is a reasonable first-order approximation of the measured CB curves.

In reality, across all workloads, we expect to see a diverse set of demand CB curves. Therefore, this approximation will be much better for some workloads than others. We address the accuracy of this approximation in Section 5.

Approximation-2 *The access $power$ of a cache is approximately $\rho\sqrt{Capacity} \times Bandwidth$.*

To quantitatively reason about energy efficiency, we must introduce a relationship between power, bandwidth, and capacity to the Moguls model. We use a first-order approximation that access power consumption to a cache level is proportional to the square root of its capacity times its bandwidth, i.e., $power \propto \sqrt{Capacity} \times Bandwidth$. The rationale for this approximation is that the power consumed in transporting data from/to memory cells dominates the total power of a cache, especially for large capacities. Therefore, two factors affect the power consumption: the data transfer rate and the transfer distance from/to the memory cells. The data transfer rate is proportional to the bandwidth, while the distance the data travels is proportional to the physical dimension of the memory array, i.e., the square root of capacity.

ρ is a constant determined by the process technology and some features of the cache design such as number of ports and number of banks. In the rest of this section, we assume ρ is constant, and the same for all levels of cache. We relax this assumption in later sections.

Figure 5(b) shows the results of this approximation. We use the well-known cache simulator CACTI [10] to collect data on a set of cache configurations with different capacities and bandwidths, but the same power. These data are represented with the so-called *iso-power curve* in CB coordinates. This iso-power curve closely matches the reference dashed line with slope $-\frac{1}{2}$, showing that this approximation is reasonable. The most deviation occurs in the tail region of the iso-power line, when the cache capacity is larger than 128MB. This is an effect of leakage power, which decreases the slope of the iso-power line as cache capacity increases. Leakage power's impact depends on its fraction of the total power, and this increases with capacity. However, the results in Section 5 show that, for bandwidth-intensive computing systems, the high bandwidth usage promises that dynamic power consumption will dominate. For SRAM caches smaller than 128MB, leakage power consumption has little impact on the quality of our iso-power line approximation. In addition, we introduce some memory technologies with relatively low leakage power in Section 4. Including these memory technologies as options when designing the cache hierarchy makes our approximation more accurate for caches with large capacities.

3.2 Cache Hierarchy Optimized for Energy-Efficiency

Using the two approximations, we can use the Moguls model to analytically derive the most energy-efficient memory hierarchy (number of cache levels and the capacity and bandwidth for each) for a given core bandwidth requirement (BR_C).

Figure 6(a) and 6(b) give an example demand CB curve defined by $BR_C(T)$ and the 2-to-$\sqrt{2}$ rule. They also show two different provided CB curves that meet the bandwidth requirements. Both have two levels: one is defined by (C_1, BP_1) and (C_2, BP_2), and the other by (C_1', BP_1') and (C_2', BP_2'). If a larger capacity first level cache ($C_1' > C_1$) is chosen, the bandwidth requirement to

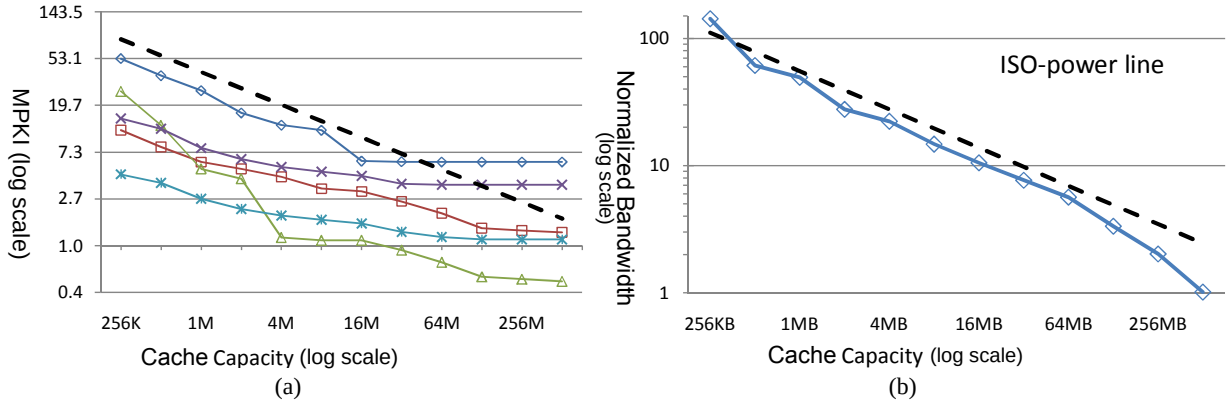

Figure 5: First-order approximations. (a) The cache miss rates per thousand instructions (MPKI) for various multi-programmed workloads under different cache capacities. (b) An iso-power line for SRAM caches (45nm process technology).

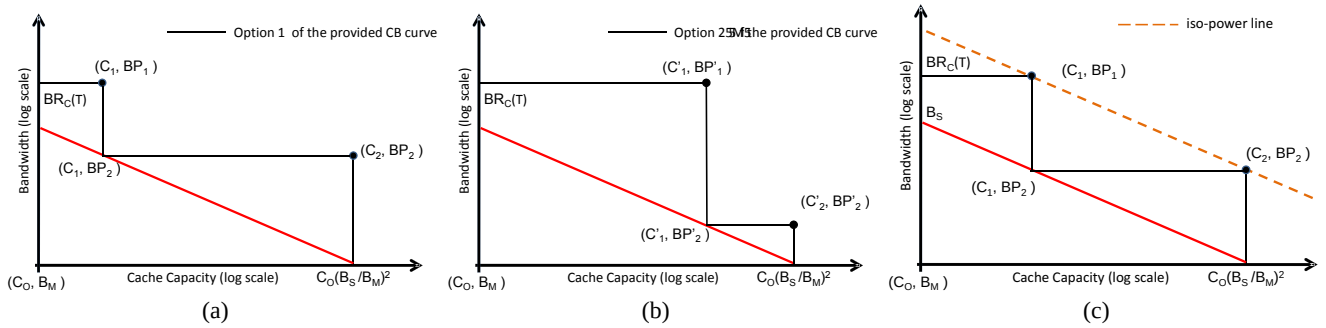

Figure 6: (a) First option of the provided CB curve based on the same demand CB curve. (b)Second option of the provided CB curve based on the same demand CB curve. (c) An optimized cache hierarchy. The two levels sit on an iso-power line.

the second level cache can be reduced ($BP_2' < BP_2$). Although both designs provide the same throughput, they may have different power consumptions.

Let us assume a two-level hierarchy, as in Figure 6(c), defined by (C_1, BP_1) and (C_2, BP_2). The total power consumption of these two levels is

$$P = \rho\sqrt{C_1}BP_1 + \rho\sqrt{C_2}BP_2 \tag{2}$$

The y-intercept of the demand CB curve is (C_O, B_S), and the x-intercept is $(C_O(\frac{B_S}{B_M})^2, B_M)$ because the slope is -0.5. The following equations are obtained based on the Moguls model,

$$BP_1 = BR_C(T); \quad C_1 = C_O(\frac{B_S}{BP_2})^2; \quad C_2 = C_O(\frac{B_S}{B_M})^2$$

If we substitute these into Eq. (2), we get

$$P = \rho\sqrt{C_O}\frac{B_S}{BP_2}BR_C(T) + \rho\sqrt{C_O}\frac{B_S}{B_M}BP_2$$

To find the BP_2 that minimize the power consumption, we need to solve $\frac{dP}{dBP_2} = 0$. We find

$$BP_2 = \sqrt{BR_C(T)B_M} \tag{3}$$

A key property is that the provided bandwidth of the second level cache is the *geometric* midpoint between $BR_C(T)$ and B_M, the bandwidth requirement of the cores and the provided bandwidth of main memory. Furthermore, both levels of the cache consume the same amount of power as $\rho\sqrt{C_O}B_S\sqrt{\frac{BR_C(T)}{B_M}}$. That is, the two levels of cache are on an iso-power line (i.e., have the same power consumption).

The results for a minimum power consumption of a two-level hierarchy can be extended to n levels of cache. Specifically, for an n-level cache design, each level has identical power consumption in order to minimize total power consumption. In other words, all points representing the caches are on the same iso-power line. Due to page constraints, we describe the induction proof verbally, as follows:

1 Assume we have n-levels of caches.
2 Assume the second to the n-th levels will be on an iso power line, but we have the flexibility to give more power or less power to the first-level cache.
3 When we give more power to the first-level cache (say, higher capacity), we can use less power for the rest of the levels to satisfy the throughput constraints. Alternatively, when we give less power to the first-level cache, we have to use more power for the rest of the levels.
4 In order to minimize the overall power, using a similar derivation to that shown earlier, we find that the first-level cache should also be on the same iso power line with the rest of the levels.

Thus, we prove our point that every level must on the same iso-power line. Furthermore, the bandwidth and capacity of each level will be evenly distributed in log-log space. The overall power consumption will be

$$P = n\rho\sqrt{C_O}B_S\sqrt[n]{\frac{BR_C(T)}{B_M}} \tag{4}$$

Based on the results of minimum power of n levels, we can also calculate the optimal levels to further improve the energy-efficiency, by setting $\frac{dP}{dn}$ to zero. The optimal number of levels

$n_{opt} = \ln \frac{BR_C(T)}{B_M}$. Note that $B_{ratio} = \frac{BR_C(T)}{B_M}$ represents the ratio of the cores' bandwidth requirement to memory's provided bandwidth. This means that as the bandwidth gap between the cores and main memory increases, more levels of cache are required to achieve the most energy-efficient cache hierarchy. This matches the historical trend, which has seen the number of levels of cache increase over time.

3.3 Throughput with Power Consumption Budget

We now add a power budget constraint to our optimization problem. While we've shown how to determine the most energy-efficient cache hierarchy that satisfies a given throughput requirement, this hierarchy may consume more than the allowed power budget. In this case, any memory hierarchy within the budget will have lower throughput than desired. The $BR_C(T)$ and the demand CB curve scale approximately linearly with throughput—as instruction throughput reduces, loads and stores execute at a lower rate. Thus, a throughput reduction results in a parallel downward shift of the demand CB curve in log-log space. Let the throughput reduction and the power budget be α and P_B, respectively. The following equation should be satisfied, in which T_{real} is the realized throughput.

$$P_B = n\rho\sqrt{C_O}(\alpha B_S)\sqrt[n]{\frac{\alpha BR_C(T)}{B_M}};$$
$$BR_C(T_{real}) = \alpha BR_C(T); \quad T_{real} = \alpha T \tag{5}$$

3.4 Throughput with Peak Bandwidth Constraint

In the previous discussion, we assumed that for a given ρ, we can build a cache with any capacity and bandwidth. In reality, the design space may be limited. Figure 7(a) shows a Moguls model plot with an extra line labeled "peak-bandwidth curve." This indicates a maximum possible bandwidth at each capacity. In the example, the most energy-efficient cache hierarchy has two levels of cache. However, the provided bandwidths of both caches exceed the peak bandwidth constraint. Thus, this optimal hierarchy cannot be built.

Figure 7(b) shows how we can provide the desired throughput by adding an extra level of cache. The extra level increases the power budget, reducing the energy efficiency of the memory system. If the power budget is violated, we must reduce the throughput. Figure 7(c) shows a sub-optimal two-level hierarchy that is within the power budget and doesn't violate the peak bandwidth constraint.

Alternatively, we can change the peak bandwidth constraint by altering the cache design or moving to a different process technology (i.e., by altering ρ). For example, the peak bandwidth of a cache can be improved by adding an access port. However, the area of the cache is greatly increased with the extra port. This indirectly increases the access energy, increasing ρ, and therefore the power consumption of the hierarchy. Thus, there is a trade-off between the peak bandwidth constraint and energy efficiency. When designing a memory hierarchy, one could feed characteristics of different cache designs into the Moguls model.

4. MEMORY HIERARCHY DESIGN WITH HYBRID TECHNOLOGIES

The factor ρ has a big impact on the optimal memory hierarchy. We've shown that for a fixed cache organization and process technology, we can approximate ρ as a fixed parameter in the Moguls model. However, in reality ρ may not be constant over the full range of capacities and bandwidths of interest. This is one reason why real memory hierarchies use multiple levels composed of different memory technologies. For example, the last level cache of Power7 is composed of embedded DRAM (eDRAM) [11]. Compared to a traditional SRAM cache, eDRAM has a slower access speed, higher density, and lower power consumption. This means that eDRAM is more energy-efficient when used as a large capacity, lower bandwidth cache—in other words, when used as a lower level cache. Recently, more emerging memory technologies such as MRAM, RRAM, and PRAM have been proposed as potential candidates for future memory hierarchy designs [12, 13, 14]. Hybrid systems employing one or more of these technologies may provide the best solution, so we consider them here.

Figure 8(a) shows iso-power lines for various memory technologies. While we form lines by connecting iso-power points from the same memory technology, *all* points on the graph represent configurations with the same power consumption. One interesting observation is that the iso-power lines cross each other. For example, iso-power lines for SRAM and MRAM cross at the point where cache capacity is about 64MB. It means that SRAM has a lower ρ than MRAM when cache capacity is less than 64MB, but MRAM has a lower ρ when the cache capacity is larger than 64MB. Because it takes more energy to access an MRAM cell than to access an SRAM cell, the power consumption of an SRAM cache is lower than that of an MRAM cache for small capacities. However, as capacity increases, the energy consumed at the wire connections becomes more important. Because MRAM has a much higher density than SRAM, the wire connections of an MRAM cache are much smaller than that of an SRAM cache, and thus consume less power. Consequently, when the capacity is large enough, the smaller energy consumption of wire connections for MRAM offsets the overhead of higher energy consumption of cell accesses and MRAM caches have a lower ρ.

The cross points of iso-power lines of different memory technologies depend not only on the process technologies and cache organization, but also on workload behavior. For example, the write access energies for PRAM and MRAM are much higher than their read access energies. For the same access intensity to a cache, if the percentage of writes is higher, the energy-efficiency of PRAM and MRAM is lower. On the contrary, for memory technologies such as SRAM and eDRAM, write operations burn a similar amount of energy as read operations. Consequently, iso-power lines for different memory technologies will cross at different points in CB space for different workloads. For the example shown in Figure 8(a), about 10% of the operations are writes.

In order to explore the benefits of using hybrid memory technologies in memory hierarchy design, we extend the Moguls model to support hybrid technologies, as shown in Figure 8(b). From the conclusion of the single level memory technology, the minimum power consumption of the left-hand-side technology and right-hand-side technology are

$$P_{min1} = n_1\rho_1\sqrt{C_O}B_S \sqrt[n_1]{\frac{BR_C(T)}{B_x}};$$
$$P_{min2} = n_2\rho_2\sqrt{C_d}B_d \sqrt[n_2]{\frac{B_x}{B_M}}, \tag{6}$$

where the left-hand-side technology has n_1 levels, the right-hand-side technology has n_2 levels, and $\sqrt{C_O}B_S = \sqrt{C_d}B_d$. The total power of the memory hierarchy is $P = P_{min1} + P_{min2}$. The goal is to find the values for n_1, n_2, and B_x that minimize overall power consumption. The procedure to solve this system is as follows:

1 Find B_x such that $\frac{dP}{dB_x} = 0$, assuming n_1 and n_2 are fixed.
2 Find n_1 such that $\frac{dP}{dn_1} = 0$, assuming $n = n_1 + n_2$.

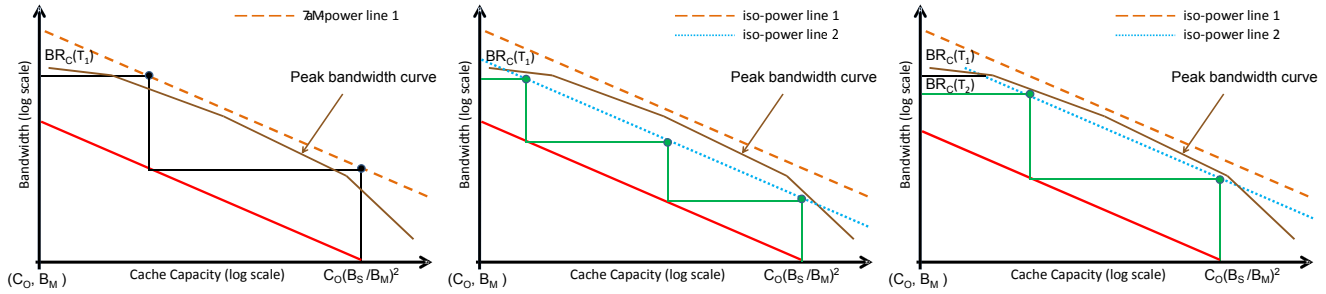

Figure 7: Moguls model with peak bandwidth constraint: (a) the provided CB curve violates the constraint; (b) the provided CB curve satisfies the constraint by adding an extra cache level; (c) the provided CB curve satisfies the constraint by degrading throughput.

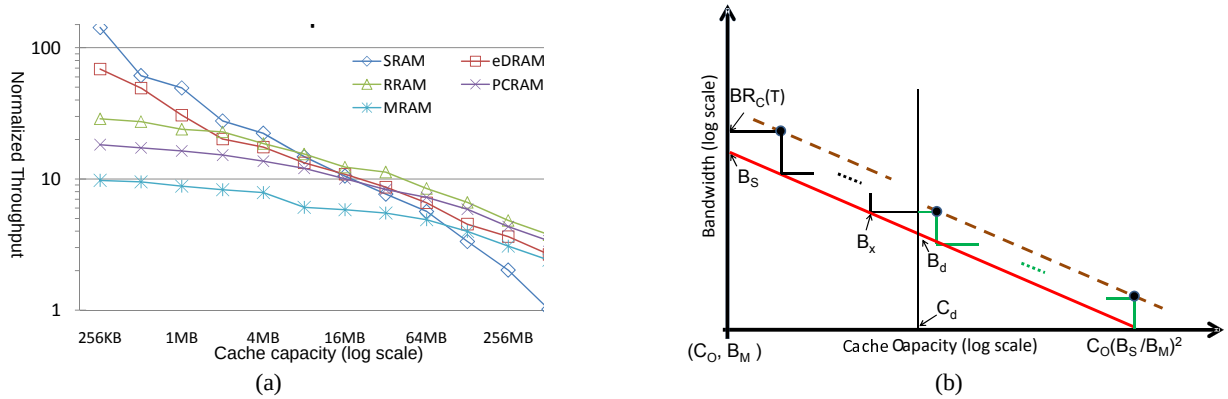

Figure 8: (a) Iso-power lines for five different memory technologies. All points consume the same power, even across technologies. (b) The Moguls model extended to hybrid memory technologies. The vertical line indicates the crossover point for the iso-power lines.

3 Find n, where $n = n_1 + n_2$, such that $\frac{dP}{dn} = 0$, by iterating the two steps above.

Intuitively, when the capacity cross-over point C_d is closer to the right hand side, n_1 tends to be larger than n_2. Conversely, when the cross-over point is closer to the left-hand-side, n_1 tends to be smaller than n_2. This is because we should implement the larger number of levels using the technology that is more efficient over a larger part of CB space.

Read/Write port:1, 64B Cache line, 4-Way	
Cache process technology	45nm
Cache rapacity range	512KB to 512MB
Memory technologies	SRAM, eDRAM, MRAM PRAM, RRAM
Level one cache	256KB (fixed)
Possible extra cache level	0 to 3
Cache policy	Inclusive

Figure 9: Cache configurations and design space.

5. EXPERIMENTS AND VALIDATION

We now validate the Moguls model in two sets of experiments. (1) We use Moguls to derive the most energy-efficient cache hierarchy for a workload, and compare that design to others via a set of exhaustive simulations. (2) We use Moguls to derive the highest throughput cache hierarchy under a fixed power budget, and compare that to the throughput of other iso-power designs nearby in the design space, again via simulation. For most workloads we evaluate, Moguls helps us derive the optimal cache hierarchy (or very close to it). In addition, we quantify the throughput improvement achieved by using multiple memory technologies.

5.1 Experimental Setup

We collect the parameters of different cache configurations from a version of CACTI that we extended to support multiple memory technologies. We also extend the cache model with a mode to process requests in a pipelined manner to estimate the peak bandwidth requirement. The detailed cache configurations are listed in Table 9. For our experiments, we focus on validating our model and methodology for a subset of the memory hierarchy between current on-die caches and main memory. We seed the memory hierarchy with a 256KB cache that can meet the cores' bandwidth requirements at that capacity. In our experiments, we consider additional levels of cache between this one and main memory, but do not alter the 256KB cache in any way. Therefore, the "core bandwidth requirement" we feed to the model is actually the bandwidth requirement of this 256KB cache. The other levels of cache are determined either by the Moguls model or are predetermined by us (e.g., when we do an exhaustive search). In practice, every additional level of cache carries overhead such as a communication interface and buffers. Consequently, the total number of cache levels should be limited. In this work, we consider designs with no more than three additional levels (i.e., besides the 256KB cache).

We use the ZESTO [15] simulator to measure performance. It is configured to model an eight-core processor. Each core is similar to Intel's Core i7. The simulator captures data addresses from all loads, stores, and prefetch operations. We use this information to calculate the memory access intensity, and use that to compute the energy consumption of the cache hierarchy. We scale the frequency of the cores to control the processor's total peak instruction throughput (i.e., assuming it is not bandwidth bound). To study

memory hierarchies for future computing systems, we evaluate processors with total instruction throughput of 32, 64, 128, and 256 billion instructions per second. The average bandwidth provided by main memory is assumed to be $8GB/S$ [16].

For our first set of experiments, we explore all reasonable points in the entire cache hierarchy design space (exhaustive simulations). That is, we consider all possible hierarchies in terms of number of levels (from 0-3 levels) and capacity of each level (256KB-512MB), although we restrict capacities to powers-of-two, and require that capacity increases as we move down the hierarchy. In these experiments, we choose the bandwidth for each cache to just meet the bandwidth requirements of the workload being evaluated. In our second set of experiments, we use the configuration determined by the Moguls model as a starting point, and evaluate a set of cache hierarchies that modify that configuration in certain ways (e.g., have twice the cache capacity). For those experiments, we carefully choose the cache bandwidths to keep all of the hierarchies at a fixed power budget. For both sets of experiments, we simulate the workloads on all candidate sets of cache hierarchies, including the Moguls-derived hierarchy, and report performance and/or power statistics.

Since we focus on a subset of the memory hierarchy close to main memory, we select workloads that are bandwidth-intensive at main memory on modern systems. Our workloads are sets of multiprogrammed benchmarks from SPEC2006 and PARSEC [17]. We randomly choose benchmarks from the full set to help us create a diverse set of demand CB curves.

Our goal has two folds: (1) to validate the Moguls model itself, and (2) to validate our methodology for applying the Moguls model to cache hierarchy design, as described in Section 3. Therefore, for our experiments when applying the Moguls model, we assume that we do not have the actual demand CB curves for our workloads, and must approximate them. Our methodology approximates the shape of a demand CB curve using the 2-to-$\sqrt{2}$ rule. We first demonstrate how well this fits our workloads.

We examined the real demand CB curves of hundreds of workloads and identified the most common patterns (i.e., shapes of the curves). Figure 10 shows examples of the two most common patterns. We also show the approximated demand CB curves (straight dash lines) for comparison. We show cache misses per thousand instructions (MPKI) vs. capacity curves—these can be converted to bandwidth vs. capacity by multiplying the MPKI by the instruction throughput and cache line size (a constant factor). The two patterns shown cover more than 90% of the workloads we examined. In pattern-1, the demand CB curve is relatively close to the line generated from the 2-to-$\sqrt{2}$ rule. In pattern-2, the real demand CB curve follows the 2-to-$\sqrt{2}$ rule quite well for low capacities. However, the MPKI levels off at some point, often due to streaming data (i.e., data accessed only once, and so not captured in even an infinite capacity cache). This pattern normally exists in workloads that have higher MPKIs, compared to those with pattern-1. The impact of different patterns on the accuracy of the Moguls model is further discussed later with experimental results. We defer discussion of workloads that fall outside these two patterns to later.

To estimate a demand CB curve, we need more than the shape of the curve. While the 2-to-$\sqrt{2}$ rule provides a first order approximation of the shape (a straight line with slope -0.5 in log-log space), we still need a starting point (e.g., y-intercept) for the curve. Further, since this exercise is intended to mimic a real design process, we should limit ourselves to a simple process to obtain the starting point. In this case study, we use a fixed 256KB cache in the memory hierarchy, and seek to design the levels between that cache and memory. Therefore, we can use the bandwidth requirement of the 256KB cache as our starting point. In other words, we need only measure the MPKI out of the 256KB cache, and then apply the 2-to-$\sqrt{2}$ rule, to estimate a workload's demand CB curve. This simple method is reasonable for designing future systems, especially when the memory design space is very large and it is not feasible to collect the real demand CB curves through simulation. Our results will validate that this simple approximation method is sufficient for our case study. We discuss other modeling methods in the next section.

Table 1: Optimized cache levels: Moguls estimation vs Simulations.

instruction throughput	Pattern-1		Pattern-2	
	Moguls	Simulation	Moguls	Simulation
32 billion/s	0	0	1	1
64 billion/s	1	1	2	2
128 billion/s	1	1	2	2
256 billion/s	2	2	3	3

5.2 Validation of Moguls Model

Our first set of experiments validates that **when optimizing for energy efficiency, the Moguls model provides the best configuration.** All configurations considered meet the throughput requirements of the processor (from 32 to 256 billion instructions per second), and thus give the same performance. However, they consume different amounts of power.

We start by looking at the number of levels in the best hierarchy vs. the hierarchy derived from Moguls. For Moguls, this number is derived from B_{ratio} (see Section 3.B), or the ratio of core bandwidth required to main memory bandwidth provided. Table 1 shows the estimated best number of levels (labeled "Moguls") and the best number found through our exhaustive simulations (labeled "Simulation"). The table shows results from a representative workload (shown in Figure 10) from each of the two key patterns of real demand CB curves.

The Moguls estimates exactly match the simulation results for the representative workloads because their real demand CB curves are close enough to the approximation we use. Further, we evaluated hundreds of workloads in this manner, and for most that fit the two key demand CB curve patterns, the results calculated by the Moguls model match well with those obtained from simulation. Overall, 92% of the workloads have an exact match between the Moguls estimate and simulation results.

We can make some additional observations. As computing throughput increases, the number of levels in the most energy-efficient hierarchy increases. This matches our expectation: increased compute throughput means a larger bandwidth gap between the core's requirements and what main memory provides; consequently, more cache levels should be added to the cache hierarchy in order to maximize energy efficiency. Similarly, the representative workload from pattern-2 requires more levels than the workload from pattern-1 because it has a larger MPKI (i.e., higher bandwidth requirement).

We next validate that the cache capacities selected by the Moguls model match those from the optimal cache hierarchy. Table 2(a) shows the detailed cache configurations for the same representative workloads. The Moguls model assumes that cache capacity is a continuous quantity. Therefore, when applying it to real system design, we need to round the cache capacities that it generates to the closest available buildable capacity. In our experiments, we assume that the capacity of cache is limited to integer powers of two[1]. Consequently, we round the Moguls-generated capacities to

[1]A real designer may have more flexibility in capacities he/she can choose, and can pick something closer to what Moguls generates.

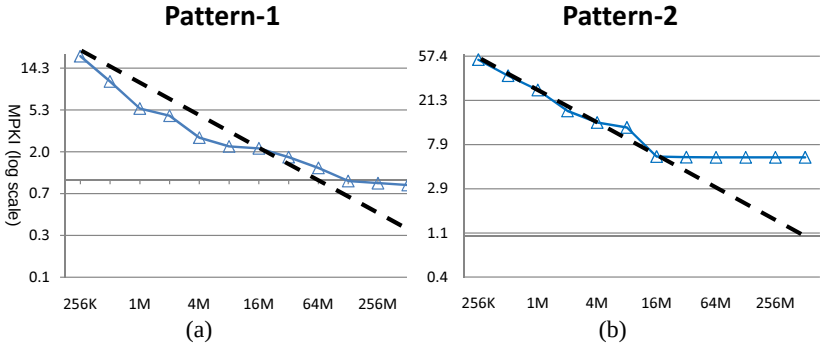

Figure 10: Two primary patterns of demanded CB curves and the approximation in Moguls. MPKI (y-axis) vs Capacity (x-axis).

Table 2: Capacity of each cache level: Moguls estimation vs Simulation results.

(a) Experimental results of workload for pattern-1

Throughput (Billion/s)	256		128		64		32	
Cache level	Moguls	Sim.	Moguls	Sim.	Moguls	Sim.	Moguls	Sim.
Level 1(MB)	1.6 (2)	2	2.8 (2)	2	0.77 (1)	1	0	0
Level 2(MB)	10.4 (8)	8	0	0	0	0	0	0

(b) Experimental results of workload for pattern-2

Throughput (Billion/s)	256		128		64		32	
Cache level	Moguls	Sim.	Moguls	Sim.	Moguls	Sim.	Moguls	Sim.
Level 1(MB)	1.04 (1)	2	1.7 (2)	2	0.84 (1)	1	0.70 (1)	1
Level 2(MB)	4.49(4)	16	12.7(16)	16	3.3(4)	4	0	0
Level 3(MB)	19.6(16)	32	0	0	0	0	0	0

the closest integer power of two (shown in the parenthesis) before comparing to the simulation results.

For the workload with pattern-1, the cache capacity results exactly match the simulation results. This is because the demand CB curve of the workload is a good fit to the Moguls approximation. For the workload with pattern-2, the cache capacities calculated by the model match the simulation results for instruction throughputs of 32, 64, and 128 billion instructions per second, but do not match at the highest instruction throughput, 256 billion instructions per second. The reason can be found in the tail region of the demand CB curve shown in Figure 10(b). The cache miss rate flattens out at a capacity of 16MB, creating a tail to the demand CB curve. For lower throughput requirements, the model still generates the optimal hierarchy because it does not attempt to pick a cache with a capacity in the tail region. However, for very high throughput requirements, the optimal hierarchy has more levels, and one of them is in the tail region. For all workloads in our study, the Moguls model achieves about 80% accuracy of calculating both optimal cache level and optimal cache capacity.

Our second set of experiments validates that **when designing under a fixed power budget, Moguls chooses the cache hierarchy with highest performance.** All cache configurations considered have the same power consumption.

We use the following procedure to generate and evaluate the various cache configurations. First, for each workload, the cache hierarchy derived from the Moguls model is adopted in the simulator. Second, we measure the maximum throughput of this cache hierarchy under the chosen power budget through simulation. Third, the cache hierarchy is replaced with different configurations. These configurations have their capacities based on the Moguls configuration, as explained below. We evaluate all possible configurations to find the maximum computing throughput for caches with those capacities under the power budget. Note that the power consumption budget needs to be carefully chosen for each workload so that the computing throughput is controlled in a reasonable range. We find the minimum power consumption required to achieve a throughput of 128 billion instructions per second for each workload, and this minimum power consumption is set as the power consumption budget.

Table 3: Cache capacities of different configurations.

	Moguls	Case1	Case2	Case3	Case4
Level 1(MB)	M1	0.5*M1	M1	2*M1	M1
Level 2(MB)	M2	M2	0.5*M2	M2	2* M2

Figure 11(a) compares the computing throughput (normalized) of the workload in Figure 10(b) for different cache configurations. The first result (labeled with "Moguls") uses the cache hierarchy derived from the Moguls model. The other cache configurations are modifications of the Moguls hierarchy. Table 3 gives the details. For our two representative workloads, the Moguls-derived configurations are the most energy efficient. Therefore, the other configurations are all lower performance—in some cases, they have much lower performance.

In Figure 11(b), the average results of computing throughput (normalized) are compared for all workloads. The first bar is the maximum computing throughput that can be achieved. Recall that we perform exhaustive simulations for each workload to determine the minimum power needed to achieve 128 billion instructions per second. Thus, our "target" throughput is 128 billion instructions per second. The second bar in the graph shows the average throughput across all workloads when using the Moguls-derived hierarchy under the power budget. On average, the Moguls-derived hierarchies achieve 92.3% of the maximum computing throughput.

5.3 The Analysis of 2-to-$\sqrt{2}$ Approximation

Although the 2-to-$\sqrt{2}$ approximation for demand CB curves fits most of our workloads, for about 8% of them, this approximation is not a good fit. These workloads fit a third pattern of demand CB curve shape. Figure 12(a) shows three example workloads that fit this pattern. As can be seen in the figure, this shape is not very

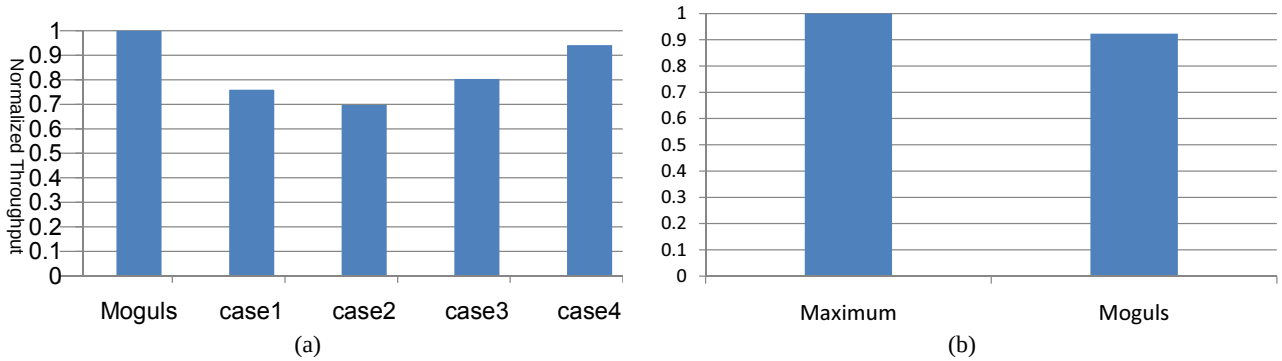

Figure 11: (a) The throughput under a power consumption budget (the example workload with pattern-2); (b) Average throughput achieved using the Moguls configuration normalized to the target throughput.

Table 4: Cache configurations.

	Moguls	O-1	O-2	O-3
Level 1(MB)	2M	512K	1M	1M
Level 2(MB)	16M	32M	16M	8M

close to a line with slope -0.5. Therefore, the demand CB curve estimated via the method in subsection 5.1 has significant error.

This shape of demand CB curve has a distinct "knee." The MPKI does not decrease much as we add cache capacity until we hit the knee. This pattern is very common in applications with a single, dominant working set—until the cache can hold the entire working set, there is little benefit to increasing capacity, and once the entire working set *does* fit, there is little benefit to any further increases in capacity. Multi-programmed workloads can also exhibit this shape of demand CB curve when their component applications have similar working set sizes. For example, if we run eight copies of the same application with the same input on a processor, it is highly possible that the real demand CB curve fits pattern-3. For a demand CB curve with this shape, we often cannot derive the optimal cache hierarchy from the Moguls model. The differences between the Moguls-derived hierarchy and the optimal hierarchy are caused by the knee. If a cache level is chosen with a capacity close to the knee, the real bandwidth requirement may be much larger than that calculated by the approximated demand CB curve. This is the source of most of the differences between the Moguls results and the best results from exhaustive simulations.

When designing real systems, we will typically not target a single workload so that the error of our approximation is mitigated. We demonstrate this experimentally as follows. Assume we want to design a system that will run all three workloads in Figure 12(a), and we want a single demand CB curve to represent all three. The figure shows the real demand CB curve for each of the three workloads. We choose our estimated demand CB curve to be a 2-to-$\sqrt{2}$ line starting from the geometric mean of the MPKIs of the three workloads at 256KB. The figure shows this as a dashed line. We then use this demand CB curve and the Moguls model to derive a cache hierarchy.

In Figure 12(b), we compare the average throughput achieved by different cache configurations across all three workloads, under a power consumption constraint. The detailed cache configurations are listed in Table 4. The label "Moguls" indicates the cache hierarchy derived from the Moguls model. "O-1", "O-2", and "O-3" represent the cache hierarchies optimized for each of the three workloads, respectively, as determined through exhaustive simulation. Despite approximating these demand CB curves with a straight line, the Moguls hierarchy is within 3% of the best hierarchy, "O-2."

5.4 Improvements from Hybrid Memory Technologies

As discussed earlier, the energy efficiency of the memory hierarchy may be improved by using multiple memory technologies. In this subsection, we evaluate the benefits of using a hybrid hierarchy. As shown in Figure 8(a), the iso-power lines of these memory technologies cross at different points in CB space, which result in different benefits.

For example. iso-power lines of SRAM and eDRAM cross at the point where the cache capacity is about 16MB. This suggests that caches larger than 16MB should be implemented with eDRAM to improve power efficiency. Using this 16MB threshold, we find that across all of our workloads, 17% of the individual caches levels should be implemented with eDRAM. We re-perform exhaustive simulations on our workloads (using the fixed power budget from earlier in this section), allowing for cache hierarchies using both SRAM and eDRAM caches. The average throughput improves by 5% when eDRAM is available.

Using the similar method, we evaluate the combination of using SRAM with MRAM/RRAM/PRAM. The results show that the throughput can be improved by 11% when RRAM is available. For MRAM and PRAM, the improvement of throughput is less than 3%. Due to the high write access energy, the caches implemented from these two technologies are more energy-efficient than those from SRAM only when the cache capacity is very large ($>$ 64MB for most workloads). In the future, as working set sizes increase with larger inputs, we believe that the benefits for hybrid memory hierarchies will also increase since many of the alternative memory technologies are energy efficient for large caches, but not small ones.

6. FUTURE WORK

We believe that the Moguls model can serve as a universal model to accelerate the memory hierarchy design process for future computing systems. We demonstrate that using some first-order approximations, the model can help designers quickly narrow down the huge set of design choices they are faced with when beginning their work. While our evaluation focuses on caches, we believe the model can be applied to other parts of the storage hierarchy, such as main memory or even hard disk drives.

There are some additional issues that, if addressed, would make the model even more valuable.

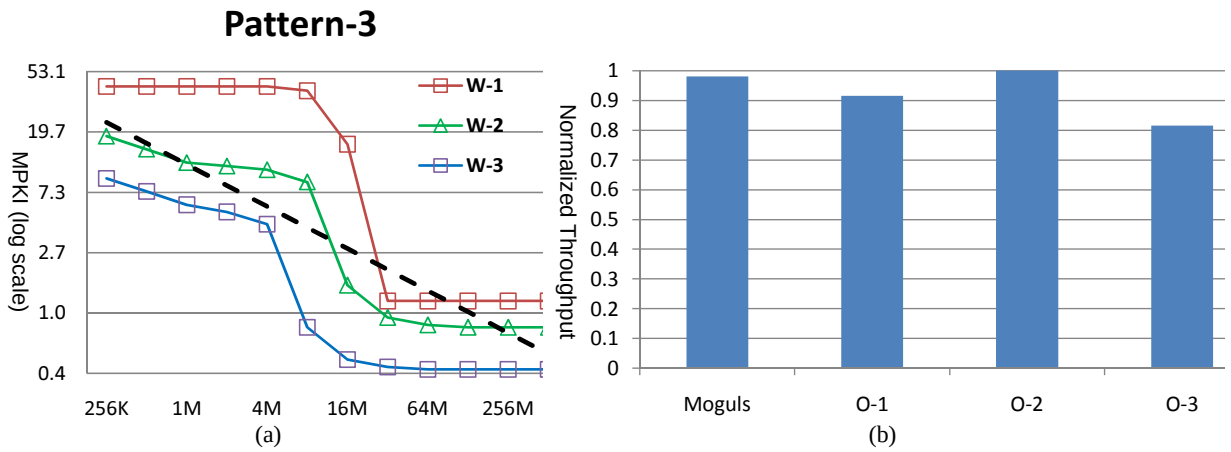

Figure 12: (a) The third pattern of demand CB curves. (b) Relative average throughput for different cache configurations.

Cost: Adding levels to a cache hierarchy may increase the number of physical components in a system (e.g., number of die or physical packages). This has a direct impact on manufacturing cost. It would be valuable to model not only a power budget for a system, but a cost budget as well.

I/O Power: Our model assumes that the relative power of different memories is determined solely by capacity and bandwidth. We ignore I/O power for off-chip memories, in part because there is so much variance among the options (e.g., die stacking vs. front-side bus).

Cache Policies: Multi-level cache hierarchies can enforce an inclusion property, an exclusion property, or something in between. Our model assumes inclusion between all levels, which greatly simplifies some of the math involved. Most (but not all) current systems enforce inclusion between all levels in the hierarchy. However, this wastes some capacity compared to an exclusion property. Also, our model assumes that all memories have the same cache line size and associativity and that all use an LRU replacement policy; alternative policies may result in slightly different provided CB curves and iso-power lines.

7. RELATED WORK

There has been much research in identifying memory bandwidth as a future performance bottleneck. In the early 90's, Ousterhout concluded that memory bandwidth contributes to non-scalability of operating systems and predicted that future machines are likely to be memory bandwidth-bound due to the speed gap between processors and memory [18]. This gap is also well cited as the "Memory Wall" [19], addressing both latency and bandwidth gaps between future processors and memories.

Burger et al. argued that memory bandwidth will become the primary memory system bottleneck due to limited off-chip bandwidth and aggressive latency hiding techniques for uniprocessor architectures [6]. Later, Huh et al. identified off-chip bandwidth as a limiter for scaling the number of cores in a CMP [20]. Other researchers in the mid-90's found that memory bandwidth limits performance for vector machines as well [21, 19]. Multiple levels of on-die cache alleviated this bandwidth problem in the past. In this paper, we argue that memory bandwidth is rapidly becoming a key performance bottleneck with the rise of bandwidth-intensive applications [22], high density compute integration, and lower bandwidth-to-compute ratio [23].

To bridge the bandwidth gap, researchers investigated adding memory levels beyond the current last-level SRAM caches. Emerging 3D chip technology enables stacking caches on top of processor cores to provide high memory bandwidth [24, 13]. Besides SRAM-based caches, new memory technologies are explored to trade off between latency, bandwidth, and cost. Wu et al. explored the power and performance implications of building an L3 cache using various memory technologies [12], including SRAM, eDRAM [25], magnetic RAM (MRAM) [13], and PRAM. These studies mainly focus on reducing the latency gap between L2 cache and external memory and do not examine bandwidth. Recently, PRAM was proposed as a DRAM alternative [26, 27] or for on-chip caches [12]. PRAM offers much higher density than DRAM with higher access latency. To mitigate the higher latency, Lee et al. proposed buffer reorganization [26], and Qureshi et al. evaluated a hybrid PRAM memory with a small DRAM buffer attached [27]. Unlike our study, they focus mainly on reducing the access time of PRAM. Most of their applications are not bandwidth-intensive, being far from bandwidth-bound. Also, those studies assume the power budget will increase to accommodate a new level of memory, while we explore what should be done within a fixed power budget.

Besides adding additional levels of memory, several architecture techniques were proposed to improve bandwidth efficiency. Compression techniques were evaluated to amplify both effective cache size [28] and effective link bandwidth [29]. Variable line size caches [30] and sectored caches [31] were proposed to increase the utilization of cache lines by bringing only useful data rather than the entire cache line. Rogers et al. summarized various bandwidth saving techniques and showed how the techniques can improve core scaling [7]. Our study focuses on analytically finding the optimal cache hierarchy; architecture techniques to save bandwidth can be applied to our model complementarily.

8. CONCLUSIONS

The bandwidth gap between processing cores and off-chip memory is increasing, especially with the emergence of throughput computing systems. Traditional cache hierarchies with only a couple levels cannot bridge the gap efficiently. In addition, constraints such as a power budget aggravate the problem. Thus, memory hierarchy design should be re-thought for future throughput computing systems. We propose a model to estimate performance of particular designs, and show how with some first-order approximations, we can use the model to quickly estimate the optimal point in a complex design space that includes multiple memory technologies.

9. REFERENCES

[1] J. Phillips, "Case study: molecular dynamics," in *Supercomputing 2007 Tutorial on High Performance Computing with CUDA*, 2007.

[2] 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*, 2007, pp. 79–87.

[3] M. Smelyanskiy, D. Holmes, J. Chhugani, A. Larson, D. M. Carmean, D. Hanson, P. Dubey, K. Augustine, D. Kim, A. Kyker, V. W. Lee, A. D. Nguyen, L. Seiler, and R. Robb, "Mapping high-fidelity volume rendering for medical imaging to CPU, GPU and many-core architectures," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1563–1570, 2009.

[4] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: a 32-way multithreaded SPARC processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, 2005.

[5] Nvidia, "Tesla C1060 datasheet," *http://www.nvidia.com/docs*, 2008.

[6] D. Burger, J. R. Goodman, and A. Kägi, "Memory bandwidth limitations of future microprocessors," in *Proceedings of the International Symposium on Computer Architecture*, 1996, pp. 78–89.

[7] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin, "Scaling the bandwidth wall: challenges in and avenues for CMP scaling," in *Proceedings of the International Symposium on Computer Architecture*, 2009, pp. 371–382.

[8] D. A. Patterson, "Latency lags bandwith," *Commununication of ACM*, vol. 47, no. 10, pp. 71–75, 2004.

[9] A. Hartstein, V. Srinivasan, T. R. Puzak, and P. G. Emma, "Cache miss behavior: is it sqrt(2)?" in *Proceedings of the Conference on Computing Frontiers*, 2006, pp. 313–320.

[10] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "CACTI 5.1 technical report HPL-2008-20," *HP Labs*, 2008.

[11] M. Ware, K. Rajamani, M. Floyd, B. Brock, J. C. Rubio, F. Rawson, and J. B. Carter, "Architecting for power management: the IBM™POWER7™approach," in *Proceedings of the International Symposium on High-Performance Computer Architecutre*, 2010, pp. 1–12.

[12] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid cache architecture with disparate memory technologies," in *Proceedings of the International Symposium on Computer Architecture*, 2009, pp. 34–45.

[13] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2009, pp. 239–249.

[14] D. L. Lewis and H.-H. S. Lee, "Architectural evaluation of 3d stacked rram caches," in *Proceedings of 3DIC*, 2009 September.

[15] G. Loh, S. Subramaniam, and Y. Xie, "Zesto: a cycle-level simulator for highly detailed microarchitecture exploration," in *Proceedings of the International Symposium on Performance Analysis of Systems and Software*, 2009, pp. 53–64.

[16] "Ddr3 sdram standard," in *http://www.jedec.org/standards-documents/docs/jesd-79-3d*, 2010 July.

[17] 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*, October 2008.

[18] J. K. Ousterhout, "Why aren't operating systems getting faster as fast as hardware?" in *Proceedings of the Summer USENIX Conference*, 1990, pp. 247–256.

[19] S. A. McKee, "Reflections on the memory wall," in *Proceedings of the Conference on Computing Frontiers*, 2004, p. 162.

[20] J. Huh, D. Burger, and S. W. Keckler, "Exploring the design space of future CMPs," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, 2001, pp. 199–210.

[21] J. D. McCalpin, "Memory bandwidth and machine balance in current high performance computers," *IEEE Technical Committee on Computer Architecture Newsletter*, 1995.

[22] W. J. Dally, "The end of denial architecture and the rise of throughput computing," in *Keynote Speech at the Design Automation Conference*, 2009.

[23] M. Reilly, "When multicore isn't enough: trends and the future for multi-multicore systems," in *Proceedings of Workshop on High Performance Embedded Computing*, 2008.

[24] N. Madan, L. Zhao, N. Muralimanohar, A. Udipi, R. Balasubramonian, R. Iyer, S. Makineni, and D. Newell, "Optimizing communication and capacity in a 3d stacked reconfigurable cache hierarchy," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2009, pp. 262–274.

[25] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi, "A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies," in *ISCA '08: Proceedings of the International Symposium on Computer Architecture*, 2008, pp. 51–62.

[26] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *Proceedings of the International Symposium on Computer Architecture*, 2009, pp. 2–13.

[27] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proceedings of the International Symposium on Computer Architecture*, 2009, pp. 24–33.

[28] M. Abrash, "A first look at the Larrabee new instructions (LRBni)," *http://www.ddj.com/hpc-high-performance-computing/216402188*, 2009.

[29] M. Thuresson, L. Spracklen, and P. Stenstrom, "Memory-link compression schemes: a value locality perspective," *IEEE Transactions on Computers*, vol. 57, no. 7, pp. 916–927, 2008.

[30] K. Inoue, K. Kai, and K. Murakami, "Dynamically variable line-size cache exploiting high on-chip memory bandwidth of merged DRAM/logic LSIs," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 1999, p. 218.

[31] A. Seznec, "Decoupled sectored caches: conciliating low tag implementation cost," in *Proceedings of the International Symposium on Computer Architecture*, 1994, pp. 384–393.