Prediction and Bounds on Shared Cache Demand from Memory Access Interleaving

Jacob Brock
University of Rochester
USA
jbrock@cs.rochester.edu

Chen Ding
University of Rochester
USA
cding@cs.rochester.edu

Rahman Lavaee*
University of Rochester
USA
rlavaee@cs.rochester.edu

Fangzhou Liu
University of Rochester
USA
fliu14@cs.rochester.edu

Liang Yuan
SKL of Computer Architecture
Institute of Computing Technology
Chinese Academy of Sciences
China
yuanliang@ict.ac.cn

Abstract

Cache in multicore machines is often shared, and the cache performance depends on how memory accesses belonging to different programs interleave with one another. The full range of performance possibilities includes all possible interleavings, which are too numerous to be studied by experiments for any mix of non-trivial programs.

This paper presents a theory to characterize the effect of memory access interleaving due to parallel execution of non-data-sharing programs. The theory uses an established metric called the footprint (which can be used to calculate miss ratios in fully-associative LRU caches) to measure cache demand, and considers the full range of interleaving possibilities. The paper proves a lower bound for footprints of interleaved traces, and then formulates an upper bound in terms of the footprints of the constituent traces. It also shows the correctness of footprint composition used in a number of existing techniques, and places precise bounds on its accuracy.

CCS Concepts • **Computing methodologies → Modeling methodologies**;

Keywords Multicore, Shared Cache, Performance Prediction, Parallel Processing, Footprint

ACM Reference Format:
Jacob Brock, Chen Ding, Rahman Lavaee, Fangzhou Liu, and Liang Yuan. 2018. Prediction and Bounds on Shared Cache Demand from Memory Access Interleaving. In *Proceedings of 2018 ACM SIGPLAN International Symposium on Memory Management (ISMM'18).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3210563.3210565

*Now at Google

ISMM'18, June 18, 2018, Philadelphia, PA, USA

ACM ISBN 978-1-4503-5801-9/18/06…$15.00
https://doi.org/10.1145/3210563.3210565

1 Introduction

There are two common approaches to analyzing cache sharing. One is co-run testing, which uses the hardware counters available on modern processors to count misses while a set of programs are running. Co-run testing, however, cannot predict how the performance would change if the group composition changes, e.g., when a new program joins or when program finishes.

The second approach is modeling. A recent example is based on the footprint, which is a measure of the average working-set size. The footprint is *composable*, which means that the joint footprint of a program group may be *computed* from the individual footprints. In the higher-order theory of locality (HOTL), the footprint is used both in composing the group effect and computing the miss ratio [1, 21]. Effectively, HOTL "composes" the miss ratio indirectly through the footprint and predicts shared cache performance for a program group without co-run testing.

Neither approach considers the full effect of interleaving. Footprint modeling assumes uniform interleaving, while the actual interleaving may be non-uniform. Co-run testing measures an actual execution. However, any instrumentation, including the reading of hardware counters, may alter an execution, and testing cannot measure the effect of its own perturbation. Even if we can model or test any particular

interleaving, we still do not know the effect of other interleavings. In real systems, the interleavings may change from execution to execution even for the same set of programs.

The number of all possible interleavings is too numerous to be studied by experiments for any mix of non-trivial programs. Stirling's approximation of factorial is $n! \approx (n/e)^n \sqrt{2\pi n}$, which we can use to approximate $\binom{2n}{n} \approx 2^{2n}$. This is the number of possible interleavings of two length-n programs. For n as small as 150, the number of possibilities approaches 3.4E88, greater than 1E80, the number of atoms estimated to exist in the known universe.

The scale and complexity of this problem warrant a theoretical study. This paper presents a set of formal theoretical results regarding the footprints that may result from independent, non-data-sharing programs interleaving memory accesses. This includes the precise effect of uniform interleavings and the lower and upper bounds for arbitrary interleavings. As stated above, when the footprint is precisely known, it can be used to calculate the miss ratio for fully associative LRU caches of any size. The precise footprint predictions in this paper can be used directly to predict shared-cache miss ratios, and the bounds on the range of possible footprints provide a starting point for future work to determine bounds on possible shared-cache miss ratios.

The rest of the paper is organized as follows. Section 2 describes the motivation for solving this problem and gives useful background information that will be used in the paper. Section 3 presents a bound for arbitrarily interleaved traces. Section 4 proposes a series of theorems regarding the footprint composition for uniformly interleaved traces. Finally, Section 5 discusses the related work, and Section 6 concludes.

2 Motivation and Background

2.1 Interference due to Interleaving

Consider two infinite length traces: "abc cba abc cba ..." and "xxx ...". In a parallel execution, the two traces are interleaved. Consider two interleaving ratios, 1:1 and 3:3, which give the following interleaved traces:

1:1 interleaving: axbxcxcxbxax ...
3:3 interleaving: abcxxxcbaxxx ...

The two interleavings have the same demand for data: any prefix whose length is a multiple of 6 contains the exact same set of data accesses in either trace. The only difference is the ratio of interleaving. However, this difference leads to different cache performance.

For a fully associative LRU cache, the following table shows the miss ratios $mr(c)$ for cache sizes $c = 2, 3$. When the cache has two blocks, the 3:3 interleaving has one miss out of every two accesses, while the 1:1 interleaving has one miss out of every three accesses.

miss ratio	interleaving 1:1	3:3
$mr(2)$	1/3	1/2
$mr(3)$	1/6	1/3

The effect is similar to one in time-sharing systems, where the cached data of the last running program is wiped out by the newly loaded program. In our example, the coarse interleaving has a greater miss ratio for a similar reason. The 3:3 ratio means three consecutive accesses each time by a program, which exerts greater interference to the peer program.

Cache interference may come from two sources. The first is *space contention*, which is an unavoidable result of cache sharing, and occurs in both partitioned[1] and unpartitioned shared caches (in partitioned caches it comes from the fact that each program is given less than the whole cache). We call the second source *interleaving loss*, which happens only in shared cache.

We can separate the two sources of interference as follows. We use 1:1 interleaving to measure the effect of space contention. Any degradation beyond that of space contention is the interleaving loss.

The interleaving loss can be measured in two distinct ways: higher miss ratio for the same cache size, and greater cache size for the same miss ratio. The preceding table shows these two types of interleaving loss. First, at the same cache size $c = 2$, interleaving loss increases the miss ratio from 1/3 to 1/2. Second, for the same miss ratio $mr(c) = 1/3$, interleaving loss increases the cache requirement from 2 to 3.

2.2 Definitions and Notation

To study the effect of interleaving, we use a measure of cache demand called the footprint. The footprint is *composable*, which means that the joint footprint of two programs may be *computed* directly from their individual footprints. Next we outline the definitions and properties related to the footprint, and the method to compute the miss ratio from the footprint. Throughout the paper we use zero-indexing. Also, metrics (such as trace length) are subscripted with the trace they refer to, except where the identity of the trace is implicit.

Definition 2.1 (Trace)**.** We use the word trace to represent a sequence of references to data or a memory location. Each reference can be an object identifier or the address of a memory cell, block, or page. The length of a trace, A, is written as N_A. The number of distinct items is written as m_A.

Definition 2.2 (Reuse Time)**.** The reuse time is the time between consecutive occurrences of the same item. E.g., in the trace *abba*, the reuse time for a is 3, and the reuse time for b is 1.

[1]We use the term partitioned for a cache that allocates a fixed (but not necessarily equal) amount of space to each program.

Definition 2.3 (Reuse Time Histogram)**.** The reuse time histogram stores the number of times each reuse time occurs in the trace. Specifically, $rt(i)$ is the number of reuses with reuse time i.

Definition 2.4 (Window)**.** The window $A(t, \tau)$ is the τ-length substring of trace A beginning at the t^{th} element[2].

Definition 2.5 (Working Set)**.** The working set of a window is the set of distinct elements in it.

Definition 2.6 (Working Set Size)**.** The working set size of a window, $A(t, \tau)$, is the size of its working set. We write this as $\omega_A(t, \tau)$.

Definition 2.7 (Disjoint Traces)**.** Two traces A and B are called disjoint if and only if no element in A is also in B and vice versa.

Definition 2.8 (Interleaving)**.** The trace $\alpha(A, B)$ is an interleaving of the disjoint traces A and B if and only if A and B are both subsequences of $\alpha(A, B)$ and $N_\alpha = N_A + N_B$.

Definition 2.9 (Footprint)**.** Xiang et al. [20] defines the footprint function, $fp(x)$, for a trace of length N. The footprint is defined for integers $x \in [0, N]$ as the average working set size among all windows of length x:

$$fp(x) \triangleq \frac{1}{N - x + 1} \sum_{t=0}^{N-x} \omega(t, x) \tag{1}$$

Definition 2.10 (Aggregate Footprint)**.** The *aggregate footprint* function is defined as only the summation term in Equation 1: $FP(x) \triangleq \sum_{t=0}^{N-x} \omega(t, x)$. Equivalently, $FP(x) = (N - x + 1)fp(x)$.

2.3 Footprint Calculation

In practice, the footprint cannot be calculated directly from Definition 2.9 because of the high cost of doing so. Xiang et al. [21] gave a linear-time algorithm based on the reuse time histogram and the first and last access times for each distinct item in the trace. Yuan et al. [23] derived another method for calculating the footprint using the same information. The aggregate footprint can be written as

$$\begin{aligned} FP(x) = xm + \sum_{i=1}^{N} rt(i) \min(i, x) - \sum_{e=1}^{m} d(x - 1 - ft(e)) \\ - \sum_{e=1}^{m} d(lt(e) - (N - x)), \end{aligned} \tag{2}$$

where $d(x) \triangleq x$ when $x > 0$ and $d(x) \triangleq 0$ otherwise, and $ft(e)$ and $lt(e)$ indicate the first and last use time of trace element e, respectively. The footprint can then be calculated from this by $fp(x) = \frac{1}{N-x+1} FP(x)$.

Equation 2 formulates the footprint using reuse and access times rather than the working set. This formulation is useful in proving some of the theorems in the paper. In addition, Equation 2 may be used in a practical implementation since its inputs, rt, ft, and lt, can be measured in time linear to the length of a trace.

From Footprint to Miss Ratio The miss ratio in a fully associative LRU cache can be computed from the footprint using the Denning-HOTL conversion [21] in two steps. Given cache size c, Denning-HOTL first finds the window length x where $fp(x) = c$, and then it takes the forward difference of the footprint at x. Using the difference operator, Δ (defined by $\Delta f(x) = f(x + 1) - f(x)$), we write this as

$$mr(c) = \Delta fp(x)\big|_{fp(x)=c}. \tag{3}$$

Previous work (which we review in Section 5) has used footprint composition without considering the precise effect of interleaving, especially non-uniform interleaving. Next we prove a set of theorems related to all interleaving.

3 Footprint Bounds for Arbitrary Interleaving

3.1 Lower Bound

In this section we determine a lower bound for the footprint of an interleaved trace, given the footprints and lengths of the two constituent traces.

Intuitively, the working set size of a window will, on average, be increased by replacing an element in the window with an element from an entirely different trace (because the supplanted element may be represented elsewhere in the window, while the newly introduced one is certainly not). This would mean that if one wishes to "interleave" two traces in such a way as to minimize the footprint of the combined trace, it would be best to keep the two traces entirely separate (i.e., to concatenate them). This intuition becomes correct for traces much longer than the window size, but it is sometimes incorrect for small traces or large window sizes. Take for example the two traces 12221 and *abbba*. For the concatenation 12221*abbba*, the aggregate footprint for length-4 windows is $FP(4) = 2 + 2 + 3 + 4 + 3 + 2 + 2 = 18$. Alternatively, for the interleaving 1*a*222*bbb*1*a*, $FP(4) = 3 + 2 + 2 + 2 + 2 + 2 + 3 = 16$. The concatenation turns out to have a larger footprint than the interleaving because of the reuse pattern at the boundary between the two concatenated traces.

In Lemma 3.1, we show that the two concatenations of A and B (AB and BA) have equal footprints for large traces. In Theorem 3.2, we show that under the same condition a concatenated trace has the smallest footprint among all possible interleavings.

Lemma 3.1. *Given two traces A and B, with lengths N_A and N_B, let $\sigma(A, B)$ be either concatenation (having length $N_\sigma \triangleq N_A + N_B$). When $x < N_A$, $x < N_B$, and $x^2 \ll N_\sigma$,*

$$fp_\sigma(x) \approx \frac{N_A}{N_\sigma} fp_A(x) + \frac{N_B}{N_\sigma} fp_B(x).$$

[2] In order to make proofs clearer, we depart from the notation of Denning [4], who uses the first parameter to indicate the *end* point of the window.

Proof. When $x < N_A$ and $x < N_B$, the aggregate footprint of the concatenated trace counts windows entirely within each constituent trace, as well as the ones at the boundary (there are $x-1$ such windows). Call the sum of the boundary windows' working sets S_σ. The smallest possible value of S_σ occurs when every boundary window contains exactly 2 distinct items. The largest occurs when every window contains x distinct items. Thus, $2(x-1) \le S_\sigma \le x(x-1)$. For either concatenation $\sigma(A,B)$, we have

$$FP_\sigma(x) = FP_A(x) + S_\sigma + FP_B(x).$$

Rewriting in terms of the footprint, this becomes

$$fp_\sigma(x) = \frac{(N_A - x + 1)fp_A(x) + S_\sigma + (N_B - x + 1)fp_B(x)}{N_\sigma - x + 1},$$

which is bounded by the possible values of S_σ. When $x \ll N_A$ and $x \ll N_B$, this is

$$fp_\sigma(x) \approx \frac{N_A}{N_\sigma} fp_A(x) + \frac{S_\sigma}{N_\sigma - x + 1} + \frac{N_B}{N_\sigma} fp_B(x).$$

And by the bounds on S_σ, when $x^2 \ll N_\sigma$ we have

$$fp_\sigma(x) \approx \frac{N_A}{N_\sigma} fp_A(x) + \frac{N_B}{N_\sigma} fp_B(x).$$

□

Theorem 3.2. *Given two traces A, B, let $\sigma(A,B)$ be either concatenation of A and B, and $\alpha(A,B)$ be any non-sequential interleaving. Let $fp_\sigma(x)$ denote the footprint of $\sigma(A,B)$ and $fp_\alpha(x)$ the footprint of $\alpha(A,B)$, and $N_\sigma \triangleq N_A + N_B$. In the limit where $x^2 \ll N_\sigma$, the footprint of the sequential trace is less than the footprint of the non-sequential one:*

$$fp_\sigma(x) < fp_\alpha(x).$$

Proof. Applying the relationship between footprint and aggregate footprint, Equation 2 may be rewritten as

$$\begin{aligned} fp(x) = \frac{xm}{N_\sigma - x + 1} &+ \frac{\sum_{i=1}^{N_\sigma} rt(i)\min(i,x)}{N_\sigma - x + 1} \\ &- \frac{\sum_{e=1}^{m} d(x - 1 - ft(e))}{N_\sigma - x + 1} \\ &- \frac{\sum_{e=1}^{m} d(lt(e) - (N_\sigma - x + 1))}{N_\sigma - x}. \end{aligned} \tag{4}$$

Consider the effect of changing from the sequential trace to the non-sequential one, on each term in Equation 4. The first term will be unchanged. In the second term, each reuse time in the non-sequential trace will be at least as large as the corresponding reuse time in the sequential trace (since there may be more items in between the uses). The effect on the reuse histogram is that $rt(i)$ will decrease by one for some value of i, and it will increase by one for some larger value of i. Thus, the second term will be larger in the non-sequential trace. The numerator in the third term is bounded by $\sum_{i=1}^{x}(x-i) = \sum_{i=1}^{x}(i-1) = \sum_{i=1}^{x-1} i = x(x-1)/2$, since each value of $ft(e)$ from 1 to x can occur at most once. By similar reasoning, the numerator of the fourth term is also bounded by $x(x-1)/2$. Thus, we have

$$fp_\alpha(x) > fp_\sigma(x) - \frac{x^2 - x}{N_\sigma - x + 1}.$$

In the limit where $x^2 \ll N_\sigma$, this approaches

$$fp_\sigma(x) < fp_\alpha(x).$$

□

3.2 Towards an Upper Bound

Non-uniform parallelism may increase the cache demand compared to uniform parallelism. In this section, we study the upper bound, which is the highest possible footprint increase from non-uniform interleaving over uniform interleaving.

We cannot yet derive the precise bound. Instead, we show that the bound lies between two bands linear in window size. We first show the lower band, by constructing two example traces and their interleaving to maximize the footprint. The bound for all programs must be at least as high as this case.

We use the two traces $D_{2:1}, D_{1:2}$ shown in Figure 1. In $D_{2:1}$, the first two thirds all access distinct data, and the last one third all access a single data block. The first part has a linear footprint, $fp(x) = x$, and the second part a constant footprint, $fp(x) = 1$. In $D_{1:2}$, the pattern is the opposite. The design resembles the patterns that form a dovetail join in carpentry. One can view the linear-footprint part as a notch and the constant-footprint part a recess.

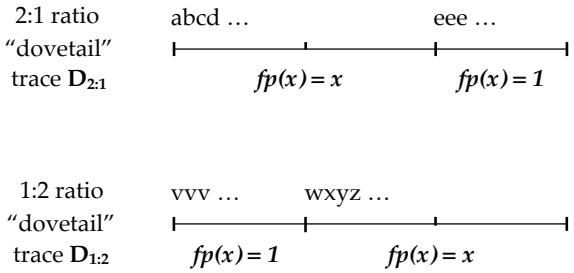

Figure 1. Two "dovetail" traces. In $D_{2:1}$, the first two thirds all access distinct data, and the last third all access a single data block. $D_{1:2}$ is the opposite.

Next we consider uniform interleavings and a special non-uniform interleaving we call "dovetail interleaving", which we denote by $\epsilon(D_{2:1}, D_{1:2})$, $\delta(D_{2:1}, D_{1:2})$ respectively.

Uniform Interleaving $\epsilon(D_{2:1}, D_{1:2})$ To compute its footprint $fp_\epsilon(x)$, we compute the interleaved footprint in each of the three parts. In the first third in each length-x window, the working-set size is $\frac{x}{2} + 1$, in which $\frac{x}{2}$ comes from $D_{2:1}$, and 1 from $D_{1:2}$. The last third is symmetrical, so it has the same footprint. In the middle third in each length-x window,

the working-set size is x, in which $\frac{x}{2}$ comes from each trace. Adding the three parts together, we have

$$fp_\epsilon(x) = \frac{2}{3}\left(\frac{x}{2}+1\right) + \frac{x}{3} = \frac{2}{3}x + \frac{2}{3}$$

where $x = 3i$ for $i \geq 1$. Note that the above calculation does not apply to windows that span between the first third and the second third and between the second third and the last third. We assume that the traces are long enough that we can ignore the effect of those windows.

Dovetail Interleaving $\delta(D_{2:1}, D_{1:2})$ We interleave the first two thirds of $D_{2:1}$ with the first third of $D_{1:2}$, uniformly at 2:1 ratio, and then interleave the remaining third of $D_{2:1}$ and the remaining two thirds of $D_{1:2}$, uniformly with 1:2 ratio. Dovetail interleaving is non-uniform, because a different mixing ratio is used in each half.

Let $fp_\delta(x)$ be the footprint of $\delta(D_{2:1}, D_{1:2})$. In each length-x window of the first half, the working-set size is $\frac{2}{3}x + 1$, in which $\frac{2}{3}x$ comes from $D_{2:1}$, and 1 from $D_{1:2}$. The second half has the same footprint because of symmetry. Therefore, we have

$$fp_\delta(x) = \frac{2}{3}x + 1$$

where $x = 3i$ for $i \geq 1$. Comparing $fp_\delta(x)$ and $fp_\delta(x)$, we see that the footprint is higher by a constant, $\frac{1}{3}$, in the dovetail-interleaving than in the uniform-interleaving. Next we extend the dovetail example to increase this difference asymptotically.

General Dovetails As mentioned earlier, we cannot yet derive the precise upper bound for the footprint of arbitrary interleaving for any group of traces (although a trivial overestimate of the bound is that $fp(x) \leq x$). Instead, we show a case where an interleaving increases $fp(x)$ over $fp_\epsilon(x)$ by $x/8$, demonstrating that the upper bound is at least $x/8$.

We adapt the dovetail example in two ways. First, instead of 2:1 ratio between the two parts of a trace, we generalize and consider k:1 ratio for $k \geq 1$. Second, the constant footprint is increased to a (for window sizes larger than a) for any $a \geq 1$.

We refer to the two dovetail traces as $D_{k:1}, D_{1:k}$. In $D_{k:1}$, the first $\frac{k}{k+1}$ of the trace all accesses distinct data, and the last $\frac{1}{k+1}$ repeatedly accesses a data blocks. In $D_{1:k}$, the pattern is the opposite.

Next are the footprint for uniform interleaving and dovetail interleaving. The calculation for the k:1 ratio traces is similar to that for the 2:1 ratio traces.

uniform $\epsilon(D_{k:1}, D_{1:k})$: $fp_\epsilon(x) = \frac{k}{k+1}x + \frac{2a}{k+1}$
dovetail $\delta(D_{k:1}, D_{1:k})$: $fp_\delta(x) = \frac{k}{k+1}x + a$

where $k \geq 1$, $a \geq 1$, $x = (k+1)ai$ for $i \geq 1$, and k, a, x, i are all integers. It is simple to verify that they compute the same footprints, derived previously for the 2:1 ratio traces, by setting $k = 2$.

For any k:1 ratio dovetails, the footprint increase is as follows:

$$fp_\delta(x) - fp_\epsilon(x) = \frac{k-1}{k+1}a$$

Two Linear Bands of Upper Bound Since it requires $x \geq (k+1)a$ for the footprints, we have $a \leq \frac{x}{k+1}$, and the difference $\frac{k-1}{k+1}a \leq \frac{k-1}{(k+1)^2}x$. The difference of $\frac{k-1}{(k+1)^2}x$ is maximized at $k = 3$, which is $\frac{x}{8}$. Hence, we have

$$fp_\delta(x) - fp_\epsilon(x) \leq \frac{x}{8}$$

Given any two traces A and B, let their uniform interleaving be $\epsilon(A, B)$. Let $\alpha(A, B)$ be any interleaving. Let $\Delta(A, B)$ be the highest possible difference, i.e.

$$fp_\alpha(x) \leq fp_\epsilon(x) + \Delta(A, B)$$

Let $\Delta_{ub} = \max\{\Delta(A, B)\}$ be the greatest increase for all programs A,B. We have the following:

Theorem 3.3. *For any group of programs, let Δ_{ub} be the greatest possible increase by arbitrary interleaving over uniform interleaving. We have:*

$$\frac{x}{8} \leq \Delta_{ub} \leq x$$

Proof. From the dovetail construction for $k = 3, a = \frac{x}{k+1} = \frac{x}{4}$, we have the pair of traces, $D_{3:1}, D_{1:3}$, whose dovetail-interleaving footprint is greater than its uniform-interleaving footprint, and the increase is as high as $\frac{x}{8}$ at $x = (k+1)a$. Since the upper bound Δ_{ub} is for all interleavings, it is at least as high as the highest increase from any given interleaving. Hence, we have $\Delta_{ub} \geq \frac{x}{8}$.

In addition, the footprint is bounded by x, i.e. $fp(\omega, x) \leq x$ for any trace ω. Let ω be any interleaved trace, we have

$$fp_\alpha(x) - fp_\epsilon(x) \leq fp_\alpha(x) \leq x$$

Hence we have the lower and the upper bands of the greatest footprint increase due to non-uniform interleaving. □

Theorem 3.3 shows that the upper bound lies between two bands that are functions linear in the window size. Although we use a special example, a pair of programs of an equal length, to derive the lower band, the theorem holds for any program groups of any length.

4 Footprint Composition of Uniform Interleaving

This section presents theorems that accurately compute or bound the effect of uniform-interleaving. The formal reasoning is made through two formulations. One is Definition 2.9, which defines the footprint as the average working-set size. We refer to it as the *working-set* based formulation. The other is Equation 2, which computes the footprint using reuse times and first and last access times. We call it *access-time* based. Table 1 shows which of the results are proved from which formulation.

Table 1. Two formulations are used to derive the footprint of uniform interleaving: based on the working set as in Definition 2.9 and on the access time as in Equation 2.

Footprint formulation	$a:b$ interleaving	1:1 interleaving	
		$fp(2x)$	$fp(2x+1)$
Working Set Based	Theorem 4.4	Corollary 4.7	
Access Time Based		Theorem 4.8	Theorem 4.9

4.1 a:b Uniform Interleaving

In this section we discuss footprint composition for a:b uniformly interleaved traces, which are composed of a elements from one trace, and b elements from another, alternating. With one correctness condition, the footprint of the interleaved trace can be calculated exactly from the footprints of the two constituent traces. We use the following definitions:

Definition 4.1 (a:b Uniform Interleaving)**.** An interleaving $\epsilon(A,B)$ of traces A and B is a:b uniform if and only if it satisfies the following properties:

1. $\alpha(A,B) = A(0,a)B(0,b)A(a,a)B(b,b)\cdots B(b(n-1),b)$. E.g., $abxcdyefz$ is a 2:1 interleaving of $abcdef$ and xyz.
2. Each window of length $p \triangleq a+b$ in $\epsilon(A,B)$ contains exactly a elements from A, and b elements from B. This property follows from the first.

Note that for a:b uniform interleavings, we use the lowercase n to represent the number of cycles. I.e., $n \triangleq N_A/a = N_B/b = N_\epsilon/p$.

Definition 4.2 (a-Stepping Footprint)**.** Given a trace of length an, for every $x \in [1,n]$ we define the *a-stepping footprint* for length-ax windows as the average working-set size of every a^{th} length-ax window, i.e., the windows which start at time ai, for any $i \in [0, n-x]$. Formally,

$$sfp(ax) \triangleq \frac{1}{n-x+1}\sum_{i=0}^{n-x}\omega(ai,ax).$$

Assumption 1 (Stepping Footprint Assumption)**.** *For a trace of length an, the a-stepping footprint for length-ax windows is approximately equal to the sliding-window footprint*[3]*.*

$$sfp(ax) \approx fp(ax)$$

Theorem 4.3. *Let $p \triangleq a+b$, and $\epsilon(A,B)$ be an a:b uniform interleaving of A and B of length pn, for which Assumption 1 holds. For all integers $x \in [1,n]$,*

$$fp_\epsilon(px) \approx fp_A(ax) + fp_B(bx) \tag{5}$$

[3]More precisely, $|sfp(ax) - fp(ax)| < a/4$. We leave the proof of this to Appendix A.

Proof. The proof follows from the definition of the aggregate footprint function. For $\epsilon(A,B)$, this is

$$FP_\epsilon(px) = \sum_{t=0}^{pn-px}\omega_\epsilon(t,px). \tag{6}$$

Let $i = \lfloor t/p \rfloor$ and $j = t \pmod p$. By the first property of a:b uniform interleaving, the working set size of $\epsilon(A,B)(t,px)$ is as follows:

$$\omega_\epsilon(t,px) = \begin{cases} \omega_A(ai+j,ax) + \omega_B(bi,bx), & 0 \le j < a \\ \omega_B(bi+j-a,bx) + \omega_A(a(i+1),ax), & a \le j < p \end{cases}$$

The summation term on the right hand side of Equation 6 is the aggregate footprint, $FP_\epsilon(px)$, and can be written as the sum of all such windows, with the last window (where $i = n-x$ and $j = 0$) written separately.

$$\begin{aligned}
FP_\epsilon(px) &= \sum_{i=0}^{n-x-1}\left(\sum_{j=0}^{a-1}\omega_A(ai+j,ax) + \omega_B(bi,bx)\right. \\
&\quad \left. + \sum_{j=a}^{p-1}\omega_B(bi+j-a,bx) + \omega_A(a(i+1),ax)\right) \\
&\quad + \omega_A(a(n-x),ax) + \omega_B(b(n-x),bx) \\
&= \omega_A(a(n-x),ax) + \sum_{i=0}^{n-x-1}\sum_{j=0}^{a-1}\omega_A(ai+j,ax) \\
&\quad + \omega_B(b(n-x),bx) + \sum_{i=0}^{n-x-1}\sum_{j=a}^{p-1}\omega_B(bi+j-a,bx) \\
&\quad + b\sum_{i=1}^{n-x}\omega_A(ai,ax) + a\sum_{i=0}^{n-x-1}\omega_B(bi,bx) \\
&= \omega_A(a(n-x),ax) + \sum_{k=0}^{an-ax-1}\omega_A(k,ax) \\
&\quad + \omega_B(b(n-x),bx) + \sum_{k=0}^{bn-bx-1}\omega_B(k,bx) \\
&\quad + b\sum_{i=1}^{n-x}\omega_A(ai,ax) + a\sum_{i=0}^{n-x-1}\omega_B(bi,bx) \\
&= \sum_{k=0}^{an-ax}\omega_A(k,ax) + \sum_{k=0}^{bn-bx}\omega_B(k,bx) \\
&\quad + b\sum_{i=1}^{n-x}\omega_A(ai,ax) + a\sum_{i=0}^{n-x-1}\omega_B(bi,bx) \\
&= (an-ax+1)fp_A(ax) + (bn-bx+1)fp_B(bx) \\
&\quad + b\sum_{i=1}^{n-x}\omega_A(ai,ax) + a\sum_{i=0}^{n-x-1}\omega_B(bi,bx).
\end{aligned} \tag{7}$$

When $n \gg x$, and by Assumption 1, we can rewrite the two summations at the end of Equation 7 based on the following:

$$\frac{1}{n-x}\sum_{i=1}^{n-x}\omega_A(ai,ax) \approx sfp_A(ax) \approx fp_A(ax)$$

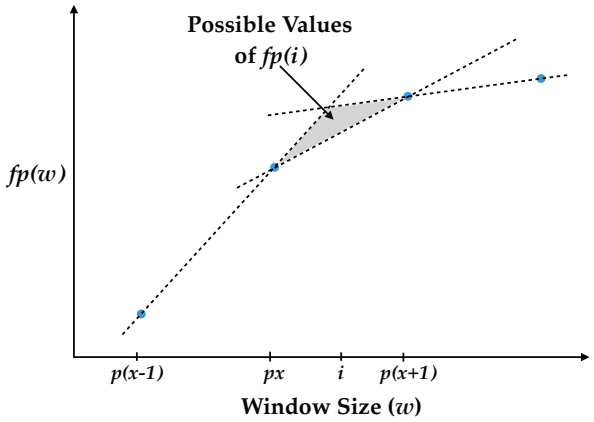

Figure 2. The concave property of footprint. The footprint for window i should lie within the shadow region

and

$$\frac{1}{n-x}\sum_{i=0}^{n-x-1}\omega_B(bi,bx) \approx sfp_B(bx) \approx fp_B(bx).$$

Thus, Equation 7 becomes

$$\begin{aligned} FP_\epsilon(px) &\approx (an-ax+1)fp_A(ax) + (bn-bx+1)fp_B(bx) \\ &\quad + (bn-bx)fp_A(ax) + (an-ax)fp_B(bx) \\ &= (pn-px+1)fp_A(ax) + (pn-xn+1)fp_B(bx) \end{aligned}$$

Dividing both sides by $(pn-px+1)$, we achieve our result:

$$fp_\epsilon(px) \approx fp_A(ax) + fp_B(bx).$$

□

Note that $fp_\epsilon(x)$ for window sizes that are non-multiples of p can be constrained using the concavity of the footprint function, which is shown by Xiang et al. [21]. We illustrate this bound in Figure 2. Consider an integer i, which is not a multiple of p. There exists some integer, x, for which $px < i < p(x+1)$. The value of $fp_\epsilon(i)$ is constrained below by the line traveling through $fp_\epsilon(px)$ and $fp_\epsilon(p(x+1))$. It is constrained above by the minimum of the line traveling through $fp_\epsilon(p(x-1))$ and $fp_\epsilon(px)$, and the line traveling through $fp_\epsilon(p(x+1))$ and $fp_\epsilon(p(x+2))$.

4.2 Error Bound for a:b Uniform Interleavings

In this section we prove a general error bound for footprint composition for uniform interleavings. The idea is to first establish a relative error bound between the footprint and the stepping footprint. Then we use that bound to prove that $fp_\epsilon(px)$ and $fp_A(ax)+fp_B(bx)$ are relatively close to each other. We formally state this result as follows:

Theorem 4.4. *Let $\epsilon(A,B)$ be an a:b uniform interleaving of A and B, respectively of lengths an and bn. With $p \triangleq a+b$, for all integers $x \in [1,n)$,*

$$\left|fp_A(ax) + fp_B(ax) - fp_\epsilon(px)\right| \le \frac{ab}{2p} + \frac{2abx}{p(n-x)} + \frac{x}{n-x}.$$

The first expression on the right hand side is a constant (only depends on a and b). The other terms are negligible when $n \gg x$.

Proof. In order to prove this bound, we first prove a lemma about how the working-set size of a window changes as it slides over the trace.

Lemma 4.5. *The working-set size of a window can change by no more than j when it slides j points over the trace. Formally,*

$$\left|\omega(t+j,a) - \omega(t,a)\right| \le j.$$

Proof. It is sufficient to prove that sliding every window one step to the right can change its working-set size by at most one. Consider the a-size window starting at time t. As this window slides to $t+1$, its working set can only change by losing the element accessed at t and acquiring the element accessed at $a+t$. Thus its cardinality may change by at most one. □

In order to prove Theorem 4.4, we first use Lemma 4.5 to find a relation between footprint and the stepping footprint for each of the traces A and B. We then combine these relations along with Equation 7 to establish the theorem.

Let us present how the aggregate footprint for trace A is related to the aggregate stepping footprint, i.e., the aggregate working set size of all stepping windows. A similar relation for trace B (which we will mention without proof) can be derived in a similiar fashion.

We partition all $an-ax+1$ length-ax windows into $n-x+1$ non-overlapping chunks, with the first and last chunks consisting, respectively, of the first $\lceil \frac{a}{2} \rceil$ and the last $\lfloor \frac{a}{2} \rfloor + 1$ windows, and each of the remaining $n-x-1$ chunks consisting of exactly a windows centerred around an a-stepping window. Specifcally, the i^{th} chunk among these chunks consists of all windows which start at a time $t \in \left[ai - \lfloor \frac{a}{2} \rfloor, ai + \lceil \frac{a}{2} \rceil\right)$.

Now within each chunk, we apply Lemma 4.5 to bound the difference between the working set size of the a-stepping window in that chunk and each of its other windows. Specifically, for every i,j where $0 \le i \le n-x$, $-\lfloor \frac{a}{2} \rfloor \le j < \lceil \frac{a}{2} \rceil$, and $0 \le ai+j < an-ax$, we have

$$|\omega_A(ai,ax) - \omega_A(ai+j,ax)| \le |j| \tag{8}$$

Summing the inequalities for every chunk (indexed by i), and all the neighbouring windows (indexed by j) in the trace

gives

$$\Bigg| \sum_{j=0}^{\lceil \frac{a}{2} \rceil - 1} \Big(\omega_A(j, ax) - \omega_A(0, ax)\Big) + \sum_{i=1}^{n-x-1} \sum_{j=-\lfloor \frac{a}{2} \rfloor}^{\lceil \frac{a}{2} \rceil - 1} \Big(\omega_A(ai + j, ax) - \omega_A(ai, ax)\Big) + \sum_{j=1}^{\lfloor \frac{a}{2} \rfloor} \Big(\omega_A(an - ax - j, ax) - \omega_A(an - ax, ax)\Big) \Bigg| \leq \sum_{j=0}^{\lceil \frac{a}{2} \rceil - 1} j + \sum_{i=1}^{n-x-1} \sum_{j=-\lfloor \frac{a}{2} \rfloor}^{\lceil \frac{a}{2} \rceil - 1} |j| + \sum_{j=0}^{\lfloor \frac{a}{2} \rfloor} j.$$

In the above inequality, the working set size of each length-ax window, except the last one, appears exactly once in the summations on the left hand side. In addition, every a-stepping window is subtracted a times by the terms in these summations (except the first and the last which are respectively subtracted $\lceil \frac{a}{2} \rceil$ and $\lfloor \frac{a}{2} \rfloor$ times). Thus, by coalescing the summations on the left hand side and evaluating the series on the right hand side of the inequality, we get

$$\begin{aligned} &\Bigg| \sum_{t=0}^{an-ax-1} \omega_A(t, ax) - a \sum_{i=1}^{n-x-1} \omega_A(ai, ax) \\ &\quad - \lceil \frac{a}{2} \rceil \omega_A(0, ax) - \lfloor \frac{a}{2} \rfloor \omega_A(an - ax, ax) \Bigg| \\ &\quad \leq (n-x) \frac{\lfloor \frac{a}{2} \rfloor (\lfloor \frac{a}{2} \rfloor + 1) + (\lceil \frac{a}{2} \rceil - 1) \lceil \frac{a}{2} \rceil}{2} \\ &\quad \leq \frac{a^2(n-x)}{4}, \end{aligned}$$

where the last inequality can be established by individually considering the even and odd cases of a.

On the left hand side of the above inequality, we subtract an extra $a \times \omega_A(an - ax, ax)$ in the second summation and add that same quantity to the last term. Then by pulling the first and last working set size terms to the right hand side we get:

$$\begin{aligned} &\Bigg| \sum_{t=0}^{an-ax-1} \omega_A(t, ax) - a \sum_{i=1}^{n-x} \omega_A(ai, ax) \Bigg| \\ &\quad \leq \frac{a^2(n-x)}{4} + \lceil \frac{a}{2} \rceil \left| \omega_A(0, ax) - \omega_A(an - ax, ax) \right|. \end{aligned}$$

The above inequality is a precise formulation of Assumption 1. On the left hand side, the first summation is the aggregate footprint over all (length-ax) windows except the last one and is equal to $FP_A(ax) - \omega_A(an - ax, ax)$. The second summation is the aggregate working set size of all a-stepping windows except the first one. The expression $\left|\omega_A(0, ax) - \omega_A(an - ax, ax)\right|$ on the right hand side is the difference between the working set size of the first and the last windows and is clearly bounded by ax. Thus multiplying both sides by $\frac{b}{a}$, we have

$$\begin{aligned} &\left| \frac{b}{a} \left(FP_A(ax) - \omega_A(an - ax, ax)\right) - b \sum_{i=1}^{n-x} \omega_A(ai, ax) \right| \\ &\quad \leq \frac{ab(n-x)}{4} + abx. \end{aligned}$$

Adding and subtracting $FP_A(ax)$ on the left hand side we get

$$\begin{aligned} &\left| \left(\frac{p}{a} FP_A(ax) - \frac{b}{a} \omega_A(an - ax, ax) \right) \right. \\ &\qquad \left. - \left(FP_A(ax) + b \sum_{i=1}^{n-x} \omega_A(ai, ax) \right) \right| \\ &\quad \leq \frac{ab(n-x)}{4} + abx. \end{aligned}$$

Replacing $FP_A(ax)$ with $(an - ax + 1) fp_A(ax)$ we obtain:

$$\begin{aligned} &\left| (pn - px + 1) fp_A(ax) - \left(FP_A(ax) + b \sum_{i=1}^{n-x} \omega_A(ai, ax) \right) \right| \\ &\quad \leq \frac{ab(n-x)}{4} + abx + \frac{b}{a} \left| fp_A(ax) - \omega(an - ax, ax) \right|. \\ &\quad \leq \frac{ab(n-x)}{4} + abx + bx. \end{aligned} \tag{9}$$

Using the same window-partitioning approach for trace B and for length-bx windows, we will obtain:

$$\begin{aligned} &\left| (pn - px + 1) fp_B(bx) - \left(FP_B(bx) + b \sum_{i=0}^{n-x-1} \omega_B(bi, bx) \right) \right| \\ &\quad \leq \frac{ab(n-x)}{4} + abx + ax. \end{aligned} \tag{10}$$

Summing up Inequalities 9 and 10, and then according to the end of Equation 7 we get

$$\begin{aligned} &\left| (pn - px + 1) \Big(fp_A(ax) + fp_B(bx) \Big) - FP_\epsilon(px) \right| \\ &\quad \leq \frac{ab(n-x)}{2} + 2abx + px. \end{aligned}$$

Finally, dividing both sides by $pn - px + 1$ gives the bound:

$$\left| fp_A(ax) + fp_B(bx) - fp_\epsilon(px) \right| \leq \frac{ab}{2p} + \frac{2abx}{p(n-x)} + \frac{x}{n-x}.$$

□

4.2.1 Tightness of the Bound

We present an example to demonstrate the tightness of the bound in Theorem 4.4 for sufficiently long traces. Intuitively, the interleaved footprint prediction is as far as possible from the actual value when the working set sizes of the a-stepping windows differ the most from their surrounding windows (and they all differ in the same direction, positive or negative).

For any k, consider a trace which consists of infinitely many repetitions of the sequence $x_1 \ldots x_k\ y_1 \ldots y_k\ x_k \ldots x_1\ y_k \ldots y_1$. When $a = k$ and $x = 2$, the sequence of working set sizes for windows of length ax will be $(2k, 2k-1, \cdots, 2k-a/2, 2k-a/2+1, \cdots 2k, \cdots)$. In fact, $\omega_A(ai, ax) - \omega_A(ai + j, ax) = j$ for all values of i and j as described in Inequality 8. Thus, when this trace is k:k uniformly interleaved with another trace of the same pattern, the footprint composition error is exactly $\frac{ab}{2p}$.

Corollary 4.6 (a:b:c... Uniform Interleaving). *Let $p \triangleq a+b+c$, let $\epsilon_2(A, B)$ be an a:b uniform interleaving of A and B, and let $\epsilon_3(A, B, C)$ be an (a+b):c uniform interleaving of $\epsilon_2(A, B)$ and C. When Assumption 1 is true, the footprint of $\epsilon_3(A, B, C)$ can be expressed for all $x \in [1, n]$ as*

$$fp_{\epsilon_3}(px) \approx fp_A(ax) + fp_B(bx) + fp_C(cx),$$

where $p \triangleq a + b + c$. Furthermore, the footprint of k uniformly interleaved traces can be expressed for all $x \in [1, n]$ as

$$fp_{\epsilon_k}(px) \approx fp_A(ax) + fp_B(bx) + fp_C(cx) + \cdots,$$

where $p \triangleq a + b + c + \cdots$.

Proof. From Theorem 4.3, an a:b uniform interleaving, $\epsilon_2(A, B)$ of A and B will have footprint $fp_{\epsilon_2}((a + b)x) \approx fp_A(ax) + fp_B(bx)$. That trace may then be (a+b):c uniformly interleaved with a third trace, C, so that $fp_{\epsilon_3}((a + b + c)x) \approx fp_\epsilon((a+b)x) + fp_C(cx) \approx fp_A(ax) + fp_B(bx) + fp_C(cx)$. Similarly, a fourth trace may be interleaved with the first three, and so on. □

Corollary 4.7 (1:1 Uniform Interleaving). *Without the need for Assumption 1, the footprint of a 1:1 uniformly interleaved trace, $\epsilon(A, B)$, composed from A and B, can be written as*

$$fp_\epsilon(2x) = fp_A(x) + fp_B(x) + \frac{fp_A(x) - \omega_A(0, x)}{2n - 2x + 1} + \frac{fp_B(x) - \omega_B(n - x, x)}{2n - 2x + 1}. \quad (11)$$

Furthermore, when $n \gg x$, it can be written in the same way as Theorem 4.3:

$$fp_\epsilon(2x) \approx fp_A(x) + fp_B(x).$$

Proof. When $a = b = 1$ (and thus, $p = 2$), Equation 7 states:

$$FP_\epsilon(2x) = (n - x + 1)fp_A(x) + (n - x + 1)fp_B(n) + \sum_{i=1}^{n-x} \omega_A(i, x) + \sum_{i=0}^{n-x-1} \omega_B(i, x)$$

By the definition of the footprint, the summation terms can each be rewritten to obtain

$$\begin{aligned} FP_\epsilon(2x) &= (n-x+1)fp_A(x) + (n-x+1)fp_B(x) + (n-x+1)fp_A(x) \\ &\quad - \omega_A(0, x) + (n - x + 1)fp_B(x) - \omega_B(n - x, x) \\ &= (2n - 2x + 1)fp_A(x) + (2n - 2x + 1)fp_B(x) \\ &\quad + fp_A(x) - \omega_A(0, x) + fp_A(x) - \omega_B(n - x, x). \end{aligned}$$

Dividing both sides by $(2n - 2x + 1)$, and applying the definitions of the footprint and aggregate footprint to the left hand side, we get

$$fp_\epsilon(2x) = fp_A(x) + fp_B(x) + \frac{fp_A(x) - \omega_A(0, x)}{2n - 2x + 1} + \frac{fp_B(x) - \omega_B(n - x, x)}{2n - 2x + 1}. \quad (12)$$

Trivially, $-x \le fp_A(x) - \omega_A(0, x) \le x$ and $-x \le fp_A(x) - \omega_B(n - x, x) \le x$. For $n \gg x$, these terms disappear, and we are left with

$$fp_\epsilon(2x) \approx fp_A(x) + fp_B(x).$$

□

Note that the last step in the proof could equivalently be taken using Assumption 1, but in this derivation it is clear how the exact form can be calculated from the working set sizes of the first window in A and the last one in B.

4.3 Access-Time Based 1:1 Uniform Interleaving

Corollary 4.7 shows how to calculate the footprint of a 1:1 uniformly interleaved trace for even-length windows. The proof followed from the working set based definition of the footprint. This approach becomes quite complicated for odd-length windows. However, using Equation 2, the derivation of the interleaved footprint for odd-length windows is more straightforward. We call this the *access time based approach*. In Theorem 4.8, we state a result that is equivalent to that of Corollary 4.7, and then in Theorem 4.9, we extend this result to cover odd-length windows. Corollary 4.10 further extends the result for interleavings of more than two traces. We prove the first theorem to illustrate the access time based approach, and omit the proof of the second theorem as it is similar.

Equation 2 shows that the footprint can be calculated from 3 attributes of a trace: (1) the first access time, $ft(e)$, for each distinct item, e, (2) its last access time, $lt(e)$, and (3) the reuse time histogram of the trace, $rt(i)$ for $i \in [1, N]$. There are simple linear relationships between each of these metrics in the constituent traces and the same metrics in the interleaved trace. The proof follows from these relationships.

Theorem 4.8. *For window size $2x$, where x is a non-negative integer, the footprint of the composed trace is*

$$FP_\epsilon(2x) = 2\Big(FP_A(x) + FP_B(x)\Big) - \sum_{e=1}^{m_A} I(ft_A(e) \le (x - 1)) - \sum_{e=1}^{m_B} I(lt_B(e) \ge (N_B - x)).$$

Note that Theorem 4.8 is equivalent to the result stated in Equation 12. This can be seen from the definitions of $FP(x)$ and $fp(x)$, and the fact that for any trace, the working set size for the first window of length x is $\omega(0, x) = \sum_{e=1}^{m_A} I((x - 1) \ge ft_A(e))$, and the working set

$$
\begin{aligned}
FP_\epsilon(2x) &= 2x(m_A+m_B) + \sum_{i=1}^{N_\epsilon} \min(i,2x) rt_\epsilon(i) - \sum_{e=1}^{m_A+m_B} d(2x-1-ft_\epsilon(e)) - \sum_{e=1}^{m_A+m_B} d(lt_\epsilon(e)-(N_\epsilon-2x)) \\
&= 2x(m_A+m_B) + \sum_{i=1}^{N_\epsilon} \min(2i,2x) rt_\epsilon(2i) - \sum_{e=1}^{m_A} d(2x-1-2ft_A(e)) \\
&\quad - \sum_{e=1}^{m_B} d(2x-2-2ft_B(e)) - \sum_{e=1}^{m_A} d(2lt_A(e)-2(N_A-x)) - \sum_{e=1}^{m_B} d(2lt_B(e)-(2N_B-2x)+1) \\
&= \left(2xm_A + \sum_{i=1}^{N_A} \min(i,x) rt_A(i) - 2\sum_{e=1}^{m_A} d(x-1-ft_A(e)) - 2\sum_{e=1}^{m_A} d(lt_A(e)-(N_A-x))\right) \\
&\quad + \left(2xm_B + \sum_{i=1}^{N_B} \min(i,x) rt_B(i) - 2\sum_{e=1}^{m_B} d(x-ft_B(e)) - 2\sum_{e=1}^{m_B} d(lt_B(e)-(N_B-x))\right) \\
&\quad - \sum_{e=1}^{m_A} I(ft_A \le (x-1)) - \sum_{e=1}^{m_B} I(lt_B(e) \ge (N_B-x)) \\
&= 2(FP_A(x)+FP_B(x)) - \sum_{e=1}^{m_A} I(ft_A(e) \le (x-1)) - \sum_{e=1}^{m_B} I(lt_B(e) \ge (N_B-x))
\end{aligned}
\tag{13}
$$

size of the last window of length x is $\omega_B(N_B - x, x) = \sum_{e=1}^{m_B} I(lt_B(e) \ge (N_B - x))$.

Proof. For a 1:1 uniformly interleaved trace $\epsilon(A, B)$, composed from traces A and B, the first and last times, and the reuse time histogram change as follows:

$$
ft_\epsilon(e) = \begin{cases} 2ft_A(e), & \text{if } e \in A \\ 2ft_B(e)+1 & \text{if } e \in B \end{cases}
$$

$$
lt_\epsilon(e) = \begin{cases} 2lt_A(e), & \text{if } e \in A \\ 2lt_B(e)+1 & \text{if } e \in B \end{cases}
$$

and

$$
rt_\epsilon(2i) = rt_A(i) + rt_B(i).
$$

From these relationships, we derive the aggregate footprint for the interleaved trace $FP_\epsilon(2x)$. We use the notation $I(P) \triangleq 1$ if the predicate P is true, and $I(P) \triangleq 0$ if it is false, and $d(x) \triangleq x \cdot I(x \ge 0)$. Note that $d(x+1) = d(x) + I(x \ge 0)$. Equation 13 shows the derivation.

□

Theorem 4.9. *For window size $2x+1$, where x is a non-negative integer, the footprint of the composed trace is*

$$
\begin{aligned}
FP_\epsilon(2x+1) &= \Big(FP_A(x+1) + FP_B(x+1) + FP_A(x) \\
&\quad + FP_B(x)\Big) - \sum_{e=1}^{m_A} I(ft_A(e) \leqslant (x-1)) \\
&\quad - \sum_{e=1}^{m_B} I(lt_B(e) \geqslant (N_B - x)).
\end{aligned}
$$

The change of window size doesn't affect the relation of *ft*, *lt* and *rt* between the interleaving trace and individual trace, and notes that for $d(x)$, we have $d(2x+1) = d(x+1)+d(x)$ and $\min(i, 2x+1) = \frac{1}{2}\big(\min(i, 2x) + \min(i, 2(x+1))\big)$. Then, we can derive the aggregate footprint $FP(2x+1)$ through the procedure similar to the proof of Theorem 4.8.

Theorem 4.8 and Theorem 4.9 can predict the aggregate footprint for two threads only. Based on these two theorems, we now extend the aggregate footprint for any number of threads that are 1:1... uniformly interleaved together.

Corollary 4.10. *For k threads, 1:1... uniformly interleaved, given any window size $kx+j$, where x is a non-negative integer, $k \ge 2$, and $0 \le j < k$, the aggregate footprint can be expressed as:*

$$
\begin{aligned}
FP_\epsilon(kx+j) &= \sum_{t=1}^{k} \Big((k-j)FP_t(x) + jFP_t(x+1)\Big) \\
&\quad - \sum_{t=1}^{k} \Big(\sum_{e=1}^{m_t} (k-t) I(ft_t(e) \le (x-1)) \\
&\quad - \sum_{e=1}^{m_t} (t-1) I(lt_t(e) \ge (N_t - x))\Big).
\end{aligned}
$$

5 Related Work

Working-Set Models of Cache The working set model was originally proposed by Denning [4]. More recent work in memory management has extended the model to include the the space-time product [6]. Denning and Schwartz [5] derived several properties of the working set, including the linear-time formula to compute the average working-set size and its conversion to the miss ratio of LRU page replacement.

Similar concepts have been used to model cache performance. To compute reuse distance, Shen et al. [17] gives a formula similar to Denning and Schwartz [5] to compute the probability that a given data element appears in a time interval of given length. Early models of cache sharing approximated the working-set size. For time-shared cache, Suh et al. [18] formulated similar properties to those of Denning

and Schwartz [5]. For multicore cache, Chandra et al. [3] defined the sequence function $seq(d, n)$ as a series of n accesses to d data blocks, where all accesses map to the same cache set. Jiang et al. [12] defined DPC as the number of distinct data blocks per cycle. Eklov and Hagersten [9] developed StatCache, whose estimate of the average reuse distance is similar to the average working-set size. We use the footprint, initially defined by Ding and Chilimbi [7]. Xiang et al. [20] gave a technique for efficiently measuring a program's footprint for all timescales. Footprint measurement has been improved by Hu et al. [11] and Yuan et al. [23].

Previous techniques show that the working-set models predict the effect of cache sharing both quickly and accurately [3, 8, 11, 12, 21]. However, those results were based on the assumption of uniform interleaving. Jiang et al. [12] predicted the increase in concurrent reuse distance by the ratio of the co-run DPC over the solo-run DPC. Brock et al. [1] proposed a general form of the footprint composition function in Theorem 4.3, as well as the *natural cache partition*, which is the average number of items kept in cache from each trace, when the traces satisfy certain properties. That work stated the footprint composition function without proof or exploration of the bounds on its accuracy for uniformly interleaved traces or the range of possible footprints for other interleavings. In this paper, we present the first study on (1) the effect of non-uniform interleaving and (2) the correctness of footprint composition for uniform interleaving. These results are applicable to other working-set models and their uses in modeling a parallel execution.

Effects of Interleaving in Shared Cache Sandberg et al. [16] studied the variation in co-run performance and showed, for example, that a program was slowed down between 1% and 17% depending on how it overlapped with another program. They developed a method to divide a program into phases and evaluate different overlappings efficiently by reusing co-run results from different phases. Instead of modeling the variation, Zhang et al. [24] neutralized it by taking the average effect by running each co-run program enough times so the difference becomes negligible in the overall performance across different tests. This method has been adopted in other studies, e.g., [20–22]. The variation observed in prior work may come from two effects: arbitrary interleaving (to create different phase overlaps) and different uniform interleaving (for the same phase overlap). Both effects are studied in this paper and given precise bounds (except for the upper bound of arbitrary interleaving).

Interleaving-based techniques have been used to improve cache sharing. Fedorova et al. [10] addressed the problem of fairness and performance isolation in shared cache and developed the *cache-fair scheduler*. The technique is based on the property that if two programs have the same frequency of cache misses, they have the same cache occupancy, so it suspends a program execution when needed to equalize the miss ratios of co-run applications. West et al. [19] developed a cache-occupancy model assuming random replacement and evaluated its accuracy in modeling LRU caches. The cache-fair scheduler effectively changes the interleaving to manage cache sharing. Zhang et al. [25] developed throttling-enabled multicore management (TEMM), which throttles the speed of a core by reducing its frequency. Scheduling and throttling are both ways to improve interleaving. The theorems in this paper can help to develop such techniques and derive their performance limits.

Much research has studied the effect of interleaving on debugging, e.g., finding races that only manifest in specific interleavings. Most methods focused on data sharing, which is less an interest for locality analysis since the actively shared data is usually cached. Burckhardt et al. [2] took a different approach and developed a method called *probabilistic concurrency testing (PCT)*. For n threads executing a total of k instructions, the interleaving generated by PCT satisfies any set of d ordering constraints with a probability at least $\frac{1}{nk^{d-1}}$. An ordering constraint restricts the choice of interleaving. Recently, Li et al. [13] used PCT to measure the variation in reuse distance with a probabilistic coverage.

Peak Memory Demand Li et al. [14, 15] defined first *average liveness* and then *peak memory demand* to measure a program's need for heap memory, which increases by memory allocation and decreases by reclamation. By demonstrating a similar mathematical formulation, they showed that heap dynamics are analogous to cache dynamics [15]. Extending this analogy, we note that the interleaving in memory allocation and reclamation by concurrent threads has an effect on the heap demand, as the interleaving of memory access does on the cache demand. In fact, the problem of interleaving is unavoidable whenever there is parallelism and (memory) resource sharing, and this effect is an important factor in performance and scalability. It remains future work whether the interleaving effect on heap demand can be predicted and bounded by extending the solutions presented in this paper.

6 Conclusions

In this paper, we explored the effects of interleaving memory accesses in shared cache, from the perspective of the footprint. This problem cannot be exhaustively addressed by co-run testing or simulation, due to the large number of possible interleavings. Our new theory first shows that for long traces, the cache demand of sequentially executed programs will always be lower than it will be for a parallel execution of the programs, and then it finds a loose bound on the footprint for any interleaving between the two programs' memory accesses.

We also proved that the footprint for an a:b uniformly interleaved trace can be calculated from the footprints of the two constituent traces, within well-defined bounds that scale with a and b.

A Appendix

Error Bound for Assumption 1

Here we demonstrate and prove an upper bound for the difference between the a-stepping footprint for windows of length ax and the footprint for a trace, which Assumption 1 claimed was small. We start with a lemma constraining how the working set size may change from one window to the next, and use this to motivate a method for maximizing the difference between the stepping footprint and the footprint.

Lemma A.1. *The working set size may change by* -1, 0, *or* $+1$ *from one window to the next. I.e.,* $\omega(t+1,x) - \omega(t,x) \in \{-1,0,1\}$.

Proof. When the window moves forward, one item departs the window, and another arrives in the window. The departure may change the working set size by either -1 (if that was the only instance of that item in the window) or 0 (otherwise). The arrival may change it by 0 (of the new item is already in the window) or 1 (otherwise). Therefore, the net change in working set size can only be -1, 0, or 1. □

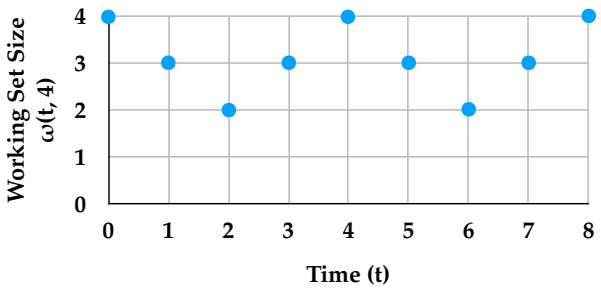

Figure 3. An example sequence of working set sizes for the trace *abcd dcba abcd*. Windows of length 4 beginning at multiples of 4 have working set sizes of 4. Between these windows, the working set size decreases as much as possible.

Theorem A.2. *For a trace of length* $N \triangleq an$, *where* a *and* n *are positive integers,*

$$\left|sfp(ax) - fp(ax)\right| < \frac{a}{4}.$$

Proof. We prove this in two parts. First we show that $sfp(ax) - fp(ax) < \frac{a}{4}$, and then we show that $fp(ax) - sfp(ax) < \frac{a}{4}$.

Call windows beginning at values of t where $t \pmod a = 0$ *a-stepping windows*. In order to find the upper bound on how much the stepping footprint can exceed the footprint, consider a trace for which $\omega(t, ax) = ax$ for all a-stepping windows. By the definition of the a-stepping footprint, $sfp(ax) = ax$. The difference $sfp(ax) - fp(ax)$ is maximized when the other windows are as much smaller than ax as possible. From Lemma A.1, it is clear that this occurs when the sequence of working set sizes for windows between consecutive a-stepping decreases by 1 and then increases by 1 until reaching the next stepping window starting point. An example of the working set sizes for such a trace (with $a = 4$ and $x = 1$) is shown in Figure 3. Another example, where $a = 4$ and $x = 3$ is $x_1 \cdots x_4\ y_1 \cdots y_4\ z_1 \cdots z_4\ x_4 \cdots x_1\ y_4 \cdots y_1\ z_4 \cdots z_1$.

Note that when a is odd, there are two consecutive windows between consecutive stepping windows with the lowest working set size of $ax - \lfloor \frac{a}{2} \rfloor$, and when a is even, there is only one window with the lowest working set size of $ax - \frac{a}{2}$. When a is odd, the aggregate footprint is then the sum of the $(n - x + 1)$ peak working set sizes, plus the sums of the working set sizes on the downslope and the upslope (of the working set size sequence as shown in Figure 3), for all $(n-x)$ cycles. When a is even, the bottom working set value occurs only on the downslope, so we count it separately, again $(n - x)$ times. The aggregate footprint is then

$$FP(ax) = \begin{cases} ax(n-x+1) + 2(n-x)\sum_{k=ax-\lfloor a/2 \rfloor}^{ax-1} k & a \text{ is odd.} \\ ax(n-x+1) + (ax - \frac{a}{2})(n-x) & \\ \quad +2(n-x)\sum_{k=ax-a/2+1}^{ax-1} k, & a \text{ is even.} \end{cases}$$

Converting to the footprint, and subtracting from the stepping footprint yields

$$sfp(ax) - fp(ax) = \begin{cases} \frac{(a^2-1)(n-x)/4}{a(n-x)+1} & a \text{ is odd.} \\ \frac{a^2(n-x)/4}{a(n-x)+1}, & a \text{ is even.} \end{cases}$$

$$< \begin{cases} \frac{a}{4} - \frac{1}{4a}, & a \text{ is odd.} \\ \frac{a}{4}, & a \text{ is even.} \end{cases}$$

In both cases, we have:

$$sfp(ax) - fp(ax) < \frac{a}{4}$$

Now we show a symmetrical result for the maximum possible value of $fp(ax) - sfp(ax)$. Consider a trace for which the stepping footprint is $ax - \lfloor a/2 \rfloor$, and the footprint climbs up to ax and back down again in between stepping windows. Two examples of such a trace are *cddc baab cddc* for $a = 4$ and $x = 1$, and $x_2x_4\ y_1 \cdots y_4\ z_1 \cdots z_4\ x_4 \cdots x_1\ y_4 \cdots y_1\ z_4 \cdots z_1\ x_1x_2$ for $a = 4$ and $x = 3$. Note that these are the same traces as used previously, with the first two . In this case, the aggregate footprint is

$$FP(ax) = \begin{cases} (ax - \lfloor a/2 \rfloor)(n-x+1) & \\ \quad +2(n-x)\sum_{k=ax-\lfloor a/2 \rfloor+1}^{ax} k & a \text{ is odd.} \\ (ax - a/2)(n-x+1) + (ax)(n-x) & \\ \quad +2(n-x)\sum_{k=ax-a/2+1}^{ax-1} k, & a \text{ is even.} \end{cases}$$

Converting to the footprint and subtracting the stepping footprint yields a familiar result:

$$fp(ax) - sfp(ax) = \begin{cases} \frac{(a^2-1)(n-x)/4}{a(n-x)+1} & a \text{ is odd.} \\ \frac{a^2(n-x)/4}{a(n-x)+1}, & a \text{ is even.} \end{cases}$$
$$< \begin{cases} \frac{a}{4} - \frac{1}{4a}, & a \text{ is odd.} \\ \frac{a}{4}, & a \text{ is even.} \end{cases}$$

Again, in both cases we have a difference smaller than $a/4$, thus, we have our result:

$$\left|sfp(ax) - fp(ax)\right| < \frac{a}{4}$$

□

Acknowledgments

The authors would like to thank Daniel Štefankovič and Rik Bose for several stimulating conversations and insights.

This work was supported by the National Science Foundation (Contract No. CCF-1717877, CCF-1629376), NFSC (No.61432018, No. 61402441, No. 61521092), National Key R&D Program of China (2017YFB0202001) and IBM CAS Faculty Fellowship.

References

[1] Jacob Brock, Chencheng Ye, Chen Ding, Yechen Li, Xiaolin Wang, and Yingwei Luo. 2015. Optimal Cache Partition-Sharing. In *International Conference on Parallel Processing*.

[2] Sebastian Burckhardt, Pravesh Kothari, Madanlal Musuvathi, and Santosh Nagarakatte. 2010. A randomized scheduler with probabilistic guarantees of finding bugs. In *ASPLOS*. 167–178.

[3] Dhruba Chandra, Fei Guo, Seongbeom Kim, and Yan Solihin. 2005. Predicting Inter-Thread Cache Contention on a Chip Multi-Processor Architecture. In *Proceedings of the International Symposium on High-Performance Computer Architecture*. 340–351.

[4] Peter J. Denning. 1968. The working set model for program behaviour. *Commun. ACM* 11, 5 (1968), 323–333.

[5] Peter J. Denning and Stuart C. Schwartz. 1972. Properties of the working set model. *Commun. ACM* 15, 3 (1972), 191–198.

[6] Peter J. Denning and Donald R. Slutz. 1978. Generalized working sets for segment reference strings. *Commun. ACM* 21, 9 (1978), 750–759.

[7] Chen Ding and Trishul Chilimbi. 2008. All-window profiling of concurrent executions. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Poster paper.*

[8] David Eklov, David Black-Schaffer, and Erik Hagersten. 2011. Fast modeling of shared caches in multicore systems. In *Proceedings of the International Conference on High Performance Embedded Architectures and Compilers*. 147–157. *Best paper.*

[9] David Eklov and Erik Hagersten. 2010. StatStack: Efficient modeling of LRU caches. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*. 55–65.

[10] Alexandra Fedorova, Margo I. Seltzer, and Michael D. Smith. 2007. Improving Performance Isolation on Chip Multiprocessors via an Operating System Scheduler. In *Proceedings of the International Conference on Parallel Architecture and Compilation Techniques*. 25–38. https://doi.org/10.1109/PACT.2007.40

[11] Xiameng Hu, Xiaolin Wang, Lan Zhou, Yingwei Luo, Chen Ding, and Zhenlin Wang. 2016. Kinetic Modeling of Data Eviction in Cache. In *Proceedings of USENIX Annual Technical Conference*. 351–364. https://www.usenix.org/conference/atc16/technical-sessions/presentation/hu

[12] Yunlian Jiang, Kai Tian, and Xipeng Shen. 2010. Combining Locality Analysis with Online Proactive Job Co-scheduling in Chip Multiprocessors. In *Proceedings of the International Conference on High Performance Embedded Architectures and Compilers*. 201–215.

[13] Hao Li, Jialiang Chang, Zijiang Yang, and Steve Carr. 2017. Memory Distance Measurement for Concurrent Programs. In *Proceedings of the Workshop on Languages and Compilers for Parallel Computing*.

[14] Pengcheng Li, Chen Ding, and Hao Luo. 2014. Modeling Heap Data Growth Using Average Liveness. In *Proceedings of the International Symposium on Memory Management*.

[15] Pengcheng Li, Hao Luo, and Chen Ding. 2016. Rethinking a heap hierarchy as a cache hierarchy: a higher-order theory of memory demand (HOTM). In *Proceedings of the International Symposium on Memory Management*. 111–121. https://doi.org/10.1145/2926697.2926708

[16] Andreas Sandberg, Andreas Sembrant, Erik Hagersten, and David Black-Schaffer. 2013. Modeling performance variation due to cache sharing. In *Proceedings of the International Symposium on High-Performance Computer Architecture*. 155–166. https://doi.org/10.1109/HPCA.2013.6522315

[17] Xipeng Shen, Jonathan Shaw, Brian Meeker, and Chen Ding. 2007. Locality approximation using time. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 55–61.

[18] G. Edward Suh, Srinivas Devadas, and Larry Rudolph. 2001. Analytical cache models with applications to cache partitioning.. In *Proceedings of the International Conference on Supercomputing*. 1–12.

[19] Richard West, Puneet Zaroo, Carl A. Waldspurger, and Xiao Zhang. 2010. Online cache modeling for commodity multicore processors. *Operating Systems Review* 44, 4 (2010), 19–29.

[20] Xiaoya Xiang, Bin Bao, Chen Ding, and Yaoqing Gao. 2011. Linear-time Modeling of Program Working Set in Shared Cache. In *Proceedings of the International Conference on Parallel Architecture and Compilation Techniques*. 350–360.

[21] Xiaoya Xiang, Chen Ding, Hao Luo, and Bin Bao. 2013. HOTL: a higher order theory of locality. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. 343–356.

[22] Chencheng Ye, Chen Ding, Hao Luo, Jacob Brock, Dong Chen, and Hai Jin. 2017. Cache Exclusivity and Sharing: Theory and Optimization. *ACM Transactions on Architecture and Code Optimization* 14, 4, 34:1–34:26. https://doi.org/10.1145/3134437

[23] Liang Yuan, Chen Ding, Peter J. Denning, and Yunquan Zhang. 2018. A Measurement Theory of Locality(MTL). *arXiv preprint arXiv:1802.01254* (2018).

[24] Xiao Zhang, Sandhya Dwarkadas, and Kai Shen. 2009. Towards practical page coloring-based multicore cache management. In *Proceedings of the EuroSys Conference*. 89–102.

[25] Xiao Zhang, Rongrong Zhong, Sandhya Dwarkadas, and Kai Shen. 2012. A Flexible Framework for Throttling-Enabled Multicore Management (TEMM). In *International Conference on Parallel Processing*. 389–398.