Multi-spectral Reuse Distance: Divining Spatial Information from Temporal Data

Anthony M. Cabrera[*][+]
Roger D. Chamberlain[*]
Jonathan C. Beard[╤]

Published in
The IEEE High Performance Extreme Computing Conference 2019 (HPEC-2019)
Boston, MA (USA)

Author affiliations and notes
[+] Work performed while on internship at Arm Inc., Austin, Texas (USA)
[*] Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO, USA
[╤] Arm Research, Austin, TX, USA

Multi-spectral Reuse Distance: Divining Spatial Information from Temporal Data

Anthony M. Cabrera*‡, Roger D. Chamberlain*, and Jonathan C. Beard†
*Dept. of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, USA
{acabrera,roger}@wustl.edu
†Arm Research, Austin, TX, USA
jonathan.beard@arm.com

***Abstract*—The problem of efficiently feeding processing elements and finding ways to reduce data movement is pervasive in computing. Efficient modeling of both temporal and spatial locality of memory references is invaluable in identifying superfluous data movement in a given application.**

To this end, we present a new way to infer both spatial and temporal locality using reuse distance analysis. This is accomplished by performing reuse distance analysis at different data block granularities: specifically, 64B, 4KiB, and 2MiB sizes. This process of simultaneously observing reuse distance with multiple granularities is called multi-spectral reuse distance. This approach allows for a qualitative analysis of spatial locality, through observing the shifting of mass in an application's reuse signature at different granularities. Furthermore, the shift of mass is empirically measured by calculating the Earth Mover's Distance between reuse signatures of an application.

From the characterization, it is possible to determine how spatially dense the memory references of an application are based on the degree to which the mass has shifted (or not shifted) and how close (or far) the Earth Mover's Distance is to zero as the data block granularity is increased. It is also possible to determine an appropriate page size from this information, and whether or not a given page is being fully utilized. From the applications profiled, it is observed that not all applications will benefit from having a larger page size. Additionally, larger data block granularities subsuming smaller ones suggest that larger pages will allow for more spatial locality exploitation, but examining the memory footprint will show whether those larger pages are fully utilized or not.

I. Introduction

At present, data movement is far more expensive than compute (i.e., an off-chip DRAM access will use $1000\times$ more energy, comparatively, than the 64-bit floating-point multiply-add that results from it when calculated using a 28nm process node [8], [16]). It follows that superfluous data movement should be reduced as much as possible as a means to improve system efficiency. Efficiently modeling the spatial and temporal locality of data has a direct impact on multiple facets of the data movement problem [18]. This includes optimal page sizing, data to memory technology placement, data page prefetching (related to placement) [27], [36], and when (and where) to use various forms of data gather/scatter. This work makes two primary contributions. First we demonstrate how to use a well known statistical technique (Earth Mover's Distance) in a novel way to inform the relationship between spatial and temporal locality. Second, we show empirically the application of our method using a set of industry standard benchmarks and how multi-spectral reuse distance analysis can inform various facets of memory management.

[‡]This work was performed while on internship at Arm Research in Austin, TX,USA.

Page sizing is often not associated with changes in data movement, though it should be. Whether using a disk controller or the main central processing unit, when data is paged-out and new data paged-in, all the contents of that page must be written to persistent storage if modified. That write-back and subsequent reloading with a new 4KiB page requires 128-256b coherence bus transactions for just one direction of movement (e.g., controller to physical DRAM). If that page isn't fully utilized once it is moved, then much of that data movement is likely wasted. Consider the case when a 2MiB page is loaded to DRAM but only half of the page is used before that physical memory is needed for another application. We will potentially have wasted 2^{15} bus transactions for loading the page, and another 2^{15} transactions (only considering wastage for the portion of the page that was not used, the full page would take 2^{16} bus transactions with a 256b bus). Even when the core is not actively participating in the transfer, cache line tag RAMs will be accessed, as will snoop/directory filters within the cache coherence network. Every access for superfluous data movement is an access taken away from useful data movement. Choosing the correct size of page is also important for copy-on-write memory systems (which most modern operating systems implement). If super (huge) pages are chosen where page utilization is low, much additional data must be copied. For example, any time a write to a child page (the virtual page pointing to a parent original page) occurs, the entire contents of that page must be copied. Simply choosing a smaller page would have been desirable. The model described in this paper could be used for online prediction for page size based on actual spatial/temporal reuse patterns, potentially with relatively low overhead.

Modern computer systems often integrate multiple memory technologies into a computer system. As an example, some GPGPU devices incorporate static random-access memory (SRAM), high bandwidth memory (HBM), and nonvolatile memory (NVM) all within the same device, and often byte addressable. The decision on where to place data within this physical memory space has direct system performance implications. Placing data on an HBM device provides very high bandwidth but intermediate latency, whereas placing data in an SRAM scratchpad could provide very low latency and high bandwidth at the expense of lower capacities (relative to other options such as NVM). Current industry practice for placing data on these memories is either to do it manually (user driven) or to treat the memory as a cache with some suitable replacement policy. The model described in this paper could be used to determine dynamically what granularity page should be used and if a predictor would be effective. Our model could do this by simplifying complex patterns, which is a side effect of the multi-spectral reuse distance approach (i.e., patterns often are easier to determine at a larger granularity versus small). Evidence presented within this work suggests that by using larger pages, the page placement prediction policy would be easier to derive due to the coarser granularity. Our model could be used as a means to decide between a caching policy or a prediction counter policy that would attempt to proactively fetch the next page.

Tightly related to data and memory technology placement is the choice of where to gather or scatter (and also compress and decompress) data. Currently the best way to decide is through extensive offline profiling on the target system. Evidence suggests that future systems will be equipped with DMA-like gather/scatter engines at multiple locations within the memory hierarchy [1], [22]. Just like the data placement decision and page sizing decisions previously mentioned, gathering data at the network interface controller (NIC) or NVM versus bringing all the data into the coherence network can pay dividends for efficiency [11]. If a system is equipped with multiple gather/scatter units, how is the system to choose between gathering at one location or another? If a reorganization function exists (provided by either the user or compiler), then using the spatial and temporal locality data provided through our described model a system could decide based on a heuristic if less data movement and tighter spatial locality could be gleaned from data reorganization.

II. Related Work

Characterization of both temporal and spatial locality has a long history [10]. Metrics from the literature include [7], [13], [17], [19], [32], [33], [34], [37].

Reuse distance–defined initially by Mattson et al. [23] as stack distance–is frequently used as a measure of temporal locality. For example, Weinberg et al. [38] define a temporal locality measure that is the area under the reuse distance curve, with the reuse distance expressed using a log scale. This formulation has been used for the characterization of various benchmarks [5], [6], [26], [29], [35]. Reuse distance has been compared with spatial locality by previous authors [12], [39]. All of these authors owe the gestalt of their works to the observations of Spirn and Denning [34] who made some of the earliest observations of program locality. Gu et al. [12] observed reuse distance to be a measure of both temporal and spatial locality. They used reuse distance as a measure of spatial locality as we do, by altering the granularity of the data block size. They reason that varying the block size leaves temporal locality unchanged, so distinctions between two block sizes are due to spatial locality. These authors also propose a spatial locality score *SLQ*. Gupta et al. [13] propose a statistical model based on the idea of "near-future windows sizes." In contrast to this work, our methodology uses Earth Mover's Distance (EMD) [30] to provide a metric that gauges spatial locality when moving histograms of multi-spectral temporal reuse data.

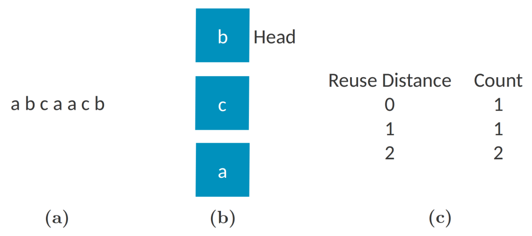
While the approach we espouse is driven by empirical data, others have taken a more theoretical approach, using the cache oblivious model to determine data locality [31] and graph theoretic approaches (interval graphs) [3]. These methods are complex, requiring (in the case of the graph approach of [3]) the search for multiple cliques over the entire stream of allocations and accesses of a program. Where these methods are intended to inform cache behavior, our methods are intended to be more general. We intend to be approximate; we feel that for many cases in real world decisions, a good fast answer is far better than a too-late exact answer.

Within this work, we make the claim that prefetching of data is a difficult problem. Mittal [24] provides an excellent overview of contemporary prefetching methods and results. Plainly speaking, the dynamic random access main memory (DRAM) of modern computers is yet another level of cache, managed by the operating system. This DRAM can be composed of many different types of memory technology, as well as having NUMA [20] characteristics. The authors make no claims of use directly as a model for prefetching, however, the proposed modeling methodology could be used to determine the optimal granularity of prefetch (in the case of memory systems) and also on the selection of cost function to drive the control process. Granularity of statistical prediction has a well known relationship with a prediction's accuracy [25] (e.g., very detailed predictions with more degrees of freedom often have more uncertainty) and we make no claim to this relationship, but we do hope that this method provides a means to more optimally use coarse grained prediction effectively (through better page sizing). The problem of data placement within a tiered and NUMA system is by no means new, and heavily related to data to disk optimization problems solved as examples in [21].

III. Methods

A. Benchmark Applications

The applications used in this work are a subset of the SPEC2006 benchmark suite [14]. However, profiling the entirety of a given benchmark proved too prohibitive. Generating reuse data for any application compiled with the *-size=train* option (i.e., the largest input size option) took several hours in the worst case. In the case of the *433.milc* benchmark compiled with the *-size=ref* option, the instrumented application took 26 days to complete. Thus, functions within this subset that



Reuse Distance	Count
0	1
1	1
2	2

Fig. 1. (a) Reference trace. (b) Reuse distance stack. (c) Reuse distance histogram.

have been shown to take a large share of the total execution time [28] were characterized. Additionally, the *MEGA-STREAM* benchmark [9], is used to demonstrate behavior of codes with very high memory access to computation ratios (itself derived from stencil computations).

Trying to save the traces of instrumented functions of the applications for post-processing also proved to be problematic because traces easily exceeded terabytes in size. Sampling reuse distances was also a possibility, but we did not want to risk aliasing a reuse distance pattern or miss unique cache line accesses. Thus, the characterization has been limited to 1 trillion references while the target function was executing.

B. Reuse Distance

In a trace of memory references, given a unique reference, its reuse distance is the number of unique references that are made before it is referenced again. Traditionally, memory references take on a cache line (64B) granularity. To calculate reuse distances for an application, a stack is employed to maintain ordering of the memory references as they are encountered. The most recently used memory reference is always at the head of the stack. There are two main operations of the reuse distance stack: encountering either new or previously seen memory references. A memory reference is added to the stack if it has not been seen during execution. When a memory reference has been encountered before, its index in the stack is isolated and the distance between its index and the head of the stack becomes the reuse distance. This reuse distance is the index into a histogram that keeps track of how many elements have a particular reuse distance. Reuse distance analysis was performed by dynamically instrumenting loads and stores using the *drcachesim* tool of *DynamoRIO* [4].

An example of calculating reuse distance is shown in Figure 1. The end result of the reuse distance analysis, i.e., the reuse distance stack in Figure 1(b) and histogram in Figure 1(c), is shown after processing the reference trace in Figure 1(a). Exploring the memory reference named a, it contributes to the reuse distance histogram as follows: the first time it is seen, it is added to the stack. The second time it is encountered, its reuse distance is calculated to be 2, and the reuse distance is 0 when it seen for the third time.

A reuse distance signature is the probability mass function (*PMF*) for the reuse distances of a given application across a range of bins. In this work, the bins represent groupings of reuse distances on a logarithmic scale.

Reuse distance analysis has traditionally been performed at cache line granularities, i.e., data blocks are set to 64B. However, our particular method uses *multi-spectral* reuse distance, which is to say that we sample reuse distance at 64B, 4KiB, and 2MiB. The 'multi-spectral' character of our methodology is what enables us to yield additional spatial locality information.

C. Earth Mover's Distance

Earth Mover's Distance (EMD) is a metric described by Rubner et al. [30] that quantifies the similarity of two histograms by finding the minimum amount of work necessary to transform the mass of one histogram into the other. In keeping with the spirit of the nomenclature, the two histograms can intuitively be viewed as a supplier and consumer of dirt (mass) that make up the two disjoint sets of a complete bipartite graph with weighted edges. The nodes of the supplier set can be viewed as piles of dirt, where the amount of earth in the pile corresponds to the value of that bin. The nodes of the consumer set can be regarded as holes, where the depth of each hole corresponds to the value of that bin. The weights are the distances between a given pile and hole. The amount of work to fill a given hole with dirt from a given pile is a function of the amount of dirt to be moved from the pile to the hole and the ground distance between the two.

More formally, bins are formed by grouping reuse distances into ranges of exponentially increasing reuse distances, with the exception of the first bin which has a range of [0,4). The bins used in this work can be observed as the labels of the x-axis in Figure 2. Mass is the value of a given bin of a reuse signature. Ground distances refer to the distance between the indices of the supplier and consumer bin. Though bin ranges grow exponentially, their indices are linear (e.g., bin with range [0,4) has index 0, bin with range [4, 8) has index 1, bin with range [8, 16) has index 2). As an example of ground distance in the context of EMD, the distance between bin [0,4) in one histogram and bin [32, 64) in the other histogram would be:

$$\begin{aligned} abs(\ \ index(\ [0,4)\ ) - index(\ [32,\ 64)\ )\ \ ) &= abs(\ 4-0\ ) \\ &= 4 \end{aligned}$$

The amount of mass located at each bin is defined by $X = x_1, \ldots, x_n$ and $Y = y_1, \ldots, y_n$, for the supplier and consumer distributions, respectively.

From this, EMD can be solved for by applying polynomial time linear programming methods [30] to minimize the following equation:

$$EMD = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} c_{ij} \tag{1}$$

where c_{ij} is the distance (cost) of moving mass from bin i to bin j and f_{ij} is the amount moved from bin i to bin j.

The minimization of EMD is subject to the following constraints:

$$f_{ij} \geq 0 \tag{2}$$

$$\sum_{j=1}^{n} f_{ij} = x_i, \quad x_i \in X \tag{3}$$

$$\sum_{i=1}^{n} f_{ij} = y_j, \quad y_j \in Y \tag{4}$$

In our case, we quantify the similarity between reuse distance signatures X and Y (e.g., reuse distance signatures for 64KiB and 4KiB granules), where f_{ij} is the amount of mass that will be moved from bin x_i to y_j and the cost of moving that mass is defined by c_{ij}. The amount of mass in both X and Y is normalized to 1, and our cost function is simply the difference between the given indices, i.e.,

$$c_{ij} = j - i$$

D. Memory Footprint

The memory footprint is derived from the final state of the reuse distance stack after performing reuse distance analysis at a given data block granularity. For each granularity, the memory footprint is calculated as follows:

$$S_{block\_granularity} \times N_{unique\_blocks} \tag{5}$$

where $S_{block\_granularity}$ is the size of the granularity used for reuse distance analysis and N_{unique_blocks} is the number of unique data blocks accessed at that granularity. Calculating the memory footprint yields a measure of how much data (in bytes) is paged in for the profiled application's region of interest.

As an example, consider the final state of the reuse distance stack in Figure 1(b). If we assume that the granularity of each block is 2MiB,

$$S_{block\_granularity} = 2MiB$$
$$N_{unique\_blocks} = 3$$
$$Memory\ Footprint = 6MiB$$

This calculation shows that 6MiB of data were paged when profiled in a given region of interest.

IV. Results and Discussion

The reuse signatures for each benchmark are shown in Figure 2. Isolating any one granularity shows typical temporal locality information such as how a particular memory subsystem will handle the memory reference access pattern of a given application (e.g., how many off-chip memory references to expect based on the *PMF* past the capacity of the last-level cache). Analyzing the reuse signatures of different granularities (a.k.a., multi-spectral reuse distance) provides valuable insight on the spatial locality of an application.

A. Spatially Dense Memory Accesses

When comparing the different signatures, there are two prototypical behaviors as the granularity of the reuse distance analysis is increased.

The first is the shift of mass in the *PMF* towards the bins of shorter reuse distances. An example of this is the result from `464.h264ref -- 2719` in Figure 2. When the granularity is 64B, almost a third of all memory references exhibit reuse distances greater than or equal to 8. At the 4KiB granularity, all memory references exhibit reuse distances no greater than 16. In the 2MiB case, virtually all reuse occurs within a reuse distance of 3.

The second behavior is the shape of the *PMF* remaining largely the same as the granularity is increased. There are two manifestations of this behavior. One is when the mass of each of the reuse signatures are contained mostly in the first bin. The result from `450.soplex -- 930` in Figure 2 shows almost identical reuse signatures for all 3 granularities, where 90% of the memory references happen within a reuse distance of 3 when the granularity is 64B, and 100% for 4KiB and 2MiB. The other manifestation is shown in the result from from `4x0.mega_stream`. For the 64B granularity, 70% of the *PMF*s mass is located in the [4,8) bin. While increasing the granularity to both 4KiB and 2MiB captures some of the mass to the right of this bin in the 64B case, the shape of the distribution remains largely unchanged.

The shifting (or not) of the *PMF* from higher to lower reuse distances bins as the granularity increases serves as a measure for how spatially dense the memory references are. A shift is indicative of memory references that reside on different data blocks at one granularity but reside on the same data block at a larger granularity. For example, refer back to the example reference trace in Section III-B and assume the granularity to be 64B. If references a, b, and c all reside on the same 4KiB data block, then when the reuse distance analysis is conducted at 4KiB granularity, then the reuse distance becomes 0 for all references. This is because the 64B data blocks that a, b, and c resided on were subsumed by the same 4KiB block. This is representative of the first prototypical behavior. If references a, b, and c reside on different 4KiB data blocks, then the reuse distances remain the same because they will not be subsumed by the same 4KiB block. Thus, we are able to observe the spatial locality for memory references by performing reuse distance analysis at different granularities.

1) Directionality of Mass Shift: Additionally, it is possible to formally prove the directionality of the mass shift that occurs when comparing the reuse signature of a smaller granularity to a larger one. In general, if we view the virtual address space of a process divorced from the physical address space underlying it, then we can view it as a contiguous space $\mathbb{A}$. Realistically this space has a natural range from 0 to $(2^{64} - 1)$ for most 64-bit architectures. Calculating the reuse distance as previously defined in Section III-B, with a single bin size of $\mathbb{A}$ would result in a distance of zero and nothing else. Consider dividing this single space $\mathbb{A}$ into two

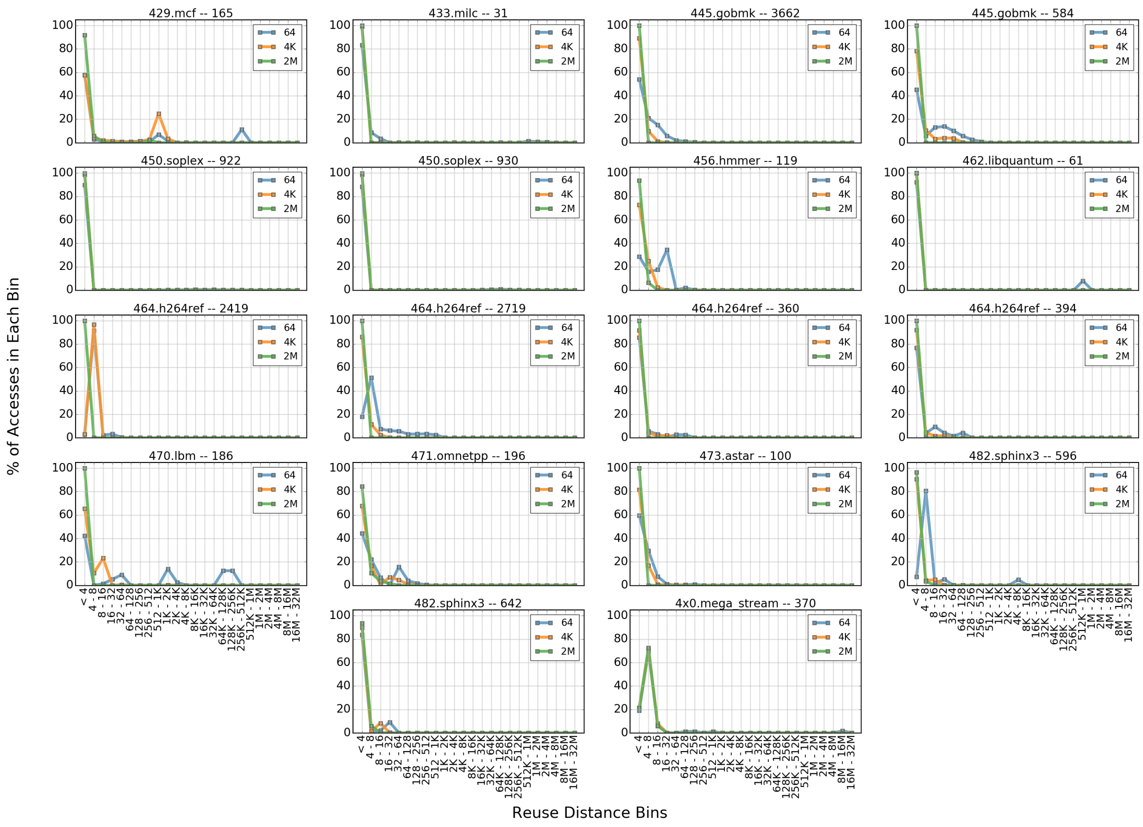


Fig. 2. Reuse distance signatures for all benchmarks. The numbers following the name of each benchmark are the line numbers on which the regions of interest for that application start.

spaces (as illustrated in Figure 3), denoted as set $\mathbb{B}$, $\frac{\mathbb{A}}{2} \rightarrow \{\mathbb{B}_0, \mathbb{B}_1\} = \mathbb{B}$. There are two spaces and two possible reuse distances: zero and one. Each of these spaces has the relation (when comparing the size of each space, or granularity of reuse bin) of: $|\mathbb{A}| > |\mathbb{B}_0| = |\mathbb{B}_1|$. It follows, then, that regardless of the the reuse bin within set $\mathbb{B}$, when superimposed over the larger set $\mathbb{A}$, the reuse distance will be zero with regards to that set. Dividing the subsets of $\mathbb{B}$ yet again yields four spaces, which we denote as set $\mathbb{C}$ corresponding to four reuse distance bins. All valid programs must fit within the space of $\mathbb{A}$. The same cannot be said of the subsets of $\mathbb{B}$ or $\mathbb{C}$. It is expected, and required, that the next larger set will subsume smaller ones. These sets are equivalent to the reuse distance granularities we have chosen, as an example, $\mathbb{B}$ could equal 2MiB, $\mathbb{C}$ could equal 4KiB, etc. If, as we have described with the multiple sized sets, we instead have multiple fixed sizes of reuse distance bins, then the bin widths should exhibit the same pattern and directionality. That is, if the distributions of each granularity are ordered with the smallest granularity bin widths in front and the largest granularity widths in back (if on a three-dimensional axis, the *PMF* of each reuse distance measurement would have the probability on the y-axis, the bin count on the x-axis, and the z-axis would be ordered from smallest to largest), then we would expect the mass when moving from front to back (with respect to the z-axis) to slide towards the zero bin of the largest granule. When ordered in this way, taking the multi-spectral reuse distance measurement has two immediate consequences we can exploit: when moving along the z-axis, we can qualitatively assess spatial density and the degree by which larger granules subsume (or do not subsume) smaller ones based on changes along the x- and y-axes. Second, with sufficiently large reuse distance bins, the mass will always converge to a zero reuse distance bin when moving in a positive direction along the z-axis (smaller reuse distance widths to larger ones).

2) *EMD as a Spatial Locality Measure:* The amount of mass that is shifted from one distribution to another is empirically shown by computing the Earth Mover's Distance between them, as described in Section III-C. The results of comparing the 64B and 4KiB distributions and the 4KiB and 2MiB ones are shown in Figure 5. The closer the EMD is to zero, the more similar the distributions are. It follows that EMDs that approach zero demonstrate behavior in which larger data block granularities do not subsume smaller ones (within the range of granularities measured, as proven previously, eventually they will always be subsumed), and that their memory reference

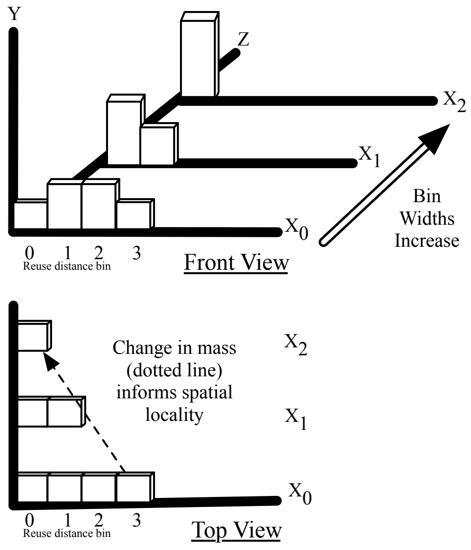


Fig. 3. Visual representation of trend described in Section IV-A1. X_0 corresponds to set $\mathbb{C}$, X_1 corresponds to set $\mathbb{B}$, X_2 corresponds to $\mathbb{A}$. The bottom graph is the view from "above" of the x and z axis showing the trend of changing mass that is expected of all applications as the bin size of each X_i approaches infinity. The rate of change in the mass (essentially slope of the line along this axis) informs the spatial locality, quantitatively measured in this work as EMD.

patterns are less spatially dense (i.e., having parts close together) than two distributions that express a large EMD.

For example, the `470.lbm -- 186` benchmark has the highest EMD score among all of the 64B vs. 4KiB comparisons. From Figure 2, at the 64B granularity, over 20% of all reuse distances are at least 4MiB away. However, we observe qualitatively in the shifting of mass from 64B to 4KiB in Figure 2, and quantitatively with Figure 5 an EMD that is much greater than zero, that much of the necessary data for computation is resident on the same 4KiB data blocks. The implications of these observations will be explored in the following subsections.

B. Page Sizing and Utilization

1) Page Sizing: The reuse signatures and their respective EMD results also have implications for selecting the page size for a given computer system. In many system architectures, it is possible to alter the page size from 4KiB or 8KiB to something larger to try and exploit spatial locality and reduce translation overhead. From the spatial locality information that results from Figures 2 and 5, it is possible to evaluate whether there are any performance benefits to increasing page size.

Referring to the `464.h264ref -- 2719` benchmark, we observe mass shifting in its reuse signatures and EMD scores that are greater than zero. In fact, at the 2MiB granularity, all of the data required for this computation is resident within strides of 0 to 8MiB, i.e., all of the mass is located in the first bin. This suggests an extremely dense spatial locality access pattern, which would benefit from larger pages.

Antithetical to this are the results from the `4x0.mega_stream -- 370`, which qualitatively in Figure 2 shows no shift in mass and has a very small EMD at all granularities. Specifically, it is shown that at least 75% of all reuse distances are occurring between 8 and 16 at all granularities. At the largest granularity, 75% of all accesses are touching data resident on at least 4 different 2MiB pages before that data is reused again. Larger page sizes are not subsuming the memory references from smaller granularities. Thus, larger page sizes cannot extract spatial locality from applications in which that spatial locality does not exist.

2) Page Utilization: The memory footprint data, calculated using 5, for each benchmark is presented in Figure 4. Each granularity is normalized to the 64B case. From this, it is possible to determine how much extraneous data, if any, is paged in when larger pages are used. When looking at Figure 4, any bar that extends past the black dotted line indicates that more memory was paged in than was necessary. We will investigate this idea further in the remainder of this section.

The `462.libquantum -- 61` benchmark results from Figures 2 and 5 show benefits for increasing larger page sizes, while also fully utilizing the data that is paged in. This is evidenced by the amount of data paged in at the 2MiB granularity being almost equal to the amount paged in for the 64B case. Referring to Equation 5, the $S_{block\_granularity}$ term will be larger in the 2MiB case than for the 64B case, but the spatially local accesses at the larger granularity decrease the N_{unique_blocks} term such that the memory footprint of the two cases are almost equal. We will now examine applications for which non-spatially local accesses result in bigger discrepancies in memory footprint at their respective measured granularities.

Looking at `464.h264ref -- 2419` and `464.h264ref -- 2719`, however, we observe that, although the 2MiB page size subsumes the smaller granules, the 2MiB page size actually pages in $10\times$ and $100\times$ more data, respective to each function, than is actually necessary, assuming that every byte of each 64B data block pulled in is fully utilized (*note:* this is a strong assumption given the previous characterizations of *Dark Bandwidth*[2]). Thus, using a 2MiB page size for this application puts undue stress on the coherence bus, and wastes a considerable amount of energy since it has to move $10\times$ and $100\times$ more data than is actually necessary.

The `4x0.mega_stream -- 370` benchmark is particularly interesting because it has been previously shown that its spatial locality access pattern is not dense, and that larger pages do not subsume the smaller data block granules and help with spatial locality. However, virtually all of the data that is paged in, even at the 2MiB granularity, is used as shown in Figure 4. Thus, the page utilization is very good for this application. This result indicates that it may be a prime candidate for a data layout transformation in order to reduce the amount of data movement and increase the amount of available physical at any given instant. The spatial

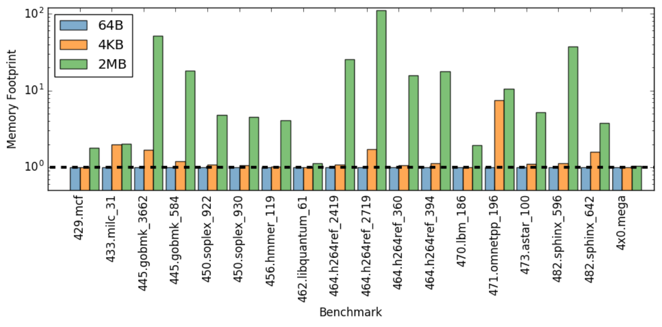


Fig. 4. Memory footprint normalized to 64B granularity.

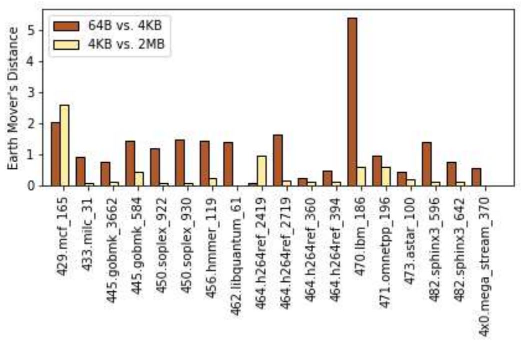


Fig. 5. Comparing (64B, 4KiB) and (4KiB, 2MiB) reuse signatures using Earth Mover's Distance.

and temporal locality patterns of this benchmark indicate that multiple values are pulled from each page at any given instant. However, streaming them in a packed fashion would improve the utilization over any given time window (recall that the overall utilization is large, but only after the entire application has executed).

C. Data Layout Transformation

The layout of the data necessary for the computation directly impacts the spatial locality characterization of an application. Recent work such as [1] shows that data movement can be reduced by transforming the layout of data near memory to better exploit spatial locality for current memory subsystem and reduce superfluous data movement. Given that a data layout transformation is possible at multiple levels of the memory hierarchy, it is possible to better determine at which level to perform the data layout transformation. We can identify the levels to perform the data layout transformation using the memory footprint analysis performed in this work.

In the case of `4x0.mega_stream`, the memory footprint data shows that, even at the largest page size, all of the data that gets paged in eventually gets used. Since even at such a large granularity the spatial access is not dense, it would be beneficial to perform the data layout transformation nearer the data, so that the data that gets paged in is densely packed, which will reduce the amount of fast physical memory that must be utilized, improve cache utilization, and lastly reduce the overall energy of computation. The last improvement would primarily be due to the reduced need to refresh DRAM rows [15] compared to a non-data layout transformation case (as less physical DRAM need be provisioned). When using a data layout transformation mechanism such as SPiDRE [1], the data could be streamed as needed potentially reducing the need to store data in DRAM.

V. Conclusions

The problem of efficiently feeding processing elements and finding ways to reduce data movement is a pervasive problem in computing. Efficient modeling of both temporal and spatial locality of memory references is invaluable in identifying superfluous data movement in a given application.

In this work, we have presented a way to model both spatial and temporal locality using what we term "multi-spectral reuse distance," derived from classic reuse distance analysis. Reuse distance is a metric traditionally used to determine the temporal locality of an application. Multi-spectral reuse distance is measured by performing reuse distance measurement at differing reuse distance granularities, in example, 64B, 4KiB, and 2MiB sizes. This approach allows for a qualitative observation of spatial locality, through observing the shifting of mass in an application's reuse signature at different granularities. Furthermore, this aspect can be quantified through the Earth Mover's Distance between ordered sets (ordered on reuse distance bin size) of probability mass functions of an application. It is these sets of *PMF*s that define the multi-spectral reuse distance. This characterization was performed on a subset of the SPEC2006 benchmark, as well as a streaming mini-application characteristic of stencil calculations.

From the multi-spectral characterization, it is possible to determine how spatially dense the memory references of an application are based on the degree to which the mass has shifted (or not shifted) and how close (or far) the Earth Mover's Distance is to zero as the data block granularity is increased. It is also possible to make inferences based on this information as to the appropriate page size, and whether or not a given page is being fully utilized. From the applications profiled, it is observed that not all applications will benefit solely from having a larger page size. Additionally, larger data block granularities subsuming smaller ones suggest that larger pages will allow for more spatial locality exploitation, but examining the memory footprint will show whether those larger pages are fully utilized or not. Finally, it is possible to infer where in the memory hierarchy a data layout transformation could be beneficial in order to more efficiently move data by observing the data utilization within given data page.

VI. Future Work

One area of future work would be to enable an automated analysis of spatial and temporal locality as a means to discern if applications would benefit from a data layout transformation on the fly, so that data layout transformations could be applied in a more demand-based way. It is a well known fact that calculating reuse distance with large bins requires storing less information than with smaller bins. This is observed by considering two cases: a reuse distance bin spanning from zero through $2^{64} - 1$ becomes just a counter while maintaining smaller bins results in more than a single counter in direct proportion to the width of the bins relative to the overall address space. A direct implication of this is that online instrumentation could be developed that exploits this property, measuring larger bins for an application while shifting to smaller bins only when necessary. A direct implication of this property, is that multi-spectral reuse distance could become a tool to address data placement and migration. Speaking more plainly, coarser granularities can be used to make the on-the-fly computation more feasible.

Acknowledgment

The authors would like to thank colleagues at Arm Research who contributed to this work through referrals to related work, helpful implementation-oriented suggestions, and spirited discussions around reuse distance. The research is supported in part by NSF grants CNS-1527510 and CNS-1763503.

References

[1] J. C. Beard, "The sparse data reduction engine (spidre): Chopping sparse data one byte at a time," in *Proc. of 2nd International Symposium on Memory Systems*. ACM, Oct. 2017.

[2] J. C. Beard and J. Randall, "Eliminating dark bandwidth: a data-centric view of scalable, efficient performance, post-moore," in *Proc. of International Conference on High Performance Computing*. Springer, 2017, pp. 106–114.

[3] M. Beg and P. Van Beek, "A graph theoretic approach to cache-conscious placement of data for direct mapped caches," *ACM SIGPLAN Notices*, vol. 45, no. 8, pp. 113–120, 2010.

[4] D. Bruening, Q. Zhao, and S. Amarasinghe, "Transparent dynamic instrumentation," *ACM SIGPLAN Notices*, vol. 47, no. 7, pp. 133–144, 2012.

[5] A. M. Cabrera, C. J. Faber, K. Cepeda, R. Derber, C. Epstein, J. Zheng, R. K. Cytron, and R. D. Chamberlain, "DIBS: A data integration benchmark suite," in *Proc. of ACM/SPEC Int'l Conf. on Performance Engineering Companion*, Apr. 2018, pp. 25–28.

[6] R. Cheveresan, M. Ramsay, C. Feucht, and I. Sharapov, "Characteristics of workloads used in high performance and technical computing," in *Proc. of ACM 21st Int'l Conf. on Supercomputing*, 2007, pp. 73–82.

[7] T. M. Conte and W.-m. W. Hwu, "Benchmark characterization for experimental system evaluation," in *Proc. of 23rd Hawaii Int'l Conf. on System Sciences*, vol. 1. IEEE, 1990, pp. 6–18.

[8] W. J. Dally. (2010) GPU Computing: To Exascale and Beyond. [Online]. Available: https://www.nvidia.com/content/PDF/sc_2010/theater/Dally_SC10.pdf

[9] T. Deakin, W. Gaudin, and S. McIntosh-Smith, "On the mitigation of cache hostile memory access patterns on many-core CPU architectures," in *Proc. of International Conference on High Performance Computing*. Springer, 2017, pp. 348–362.

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

[11] M. Gokhale, B. Holmes, and K. Iobst, "Processing in memory: The Terasys massively parallel PIM array," *Computer*, vol. 28, no. 4, pp. 23–31, 1995.

[12] X. Gu, I. Christopher, T. Bai, C. Zhang, and C. Ding, "A component model of spatial locality," in *Proc. of ACM Int'l Symp. on Memory Management*, 2009, pp. 99–108.

[13] S. Gupta, P. Xiang, Y. Yang, and H. Zhou, "Locality principle revisited: A probability-based quantitative approach," *Journal of Parallel and Distributed Computing*, vol. 73, no. 7, pp. 1011–1027, 2013.

[14] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[15] B. Jacob, S. Ng, and D. Wang, *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann, 2010.

[16] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Quantifying the energy cost of data movement in scientific applications," in *Proc. of IEEE Int'l Symp. on Workload Characterization*, Sep. 2013, pp. 56–65.

[17] S. Kumar and C. Wilkerson, "Exploiting spatial locality in data caches using spatial footprints," in *Proc. of 25th Int'l Symp. on Computer Architecture*, 1998, pp. 357–368.

[18] G. Kurian, O. Khan, and S. Devadas, "The locality-aware adaptive cache coherence protocol," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 523–534, 2013.

[19] T. Lafage and A. Seznec, "Choosing representative slices of program execution for microarchitecture simulations: A preliminary application to the data stream," in *Workload Characterization of Emerging Computer Applications*, ser. SECS, L. K. John and A. M. G. Maynard, Eds., 2001, vol. 610, pp. 145–163.

[20] C. Lameter, "NUMA (non-uniform memory access): An overview," *Queue*, vol. 11, no. 7, p. 40, 2013.

[21] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis using Queueing Network Models*. Prentice-Hall, Inc., 1984.

[22] S. Lloyd and M. Gokhale, "In-memory data rearrangement for irregular, data-intensive computing," *Computer*, no. 8, pp. 18–25, 2015.

[23] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation techniques for storage hierarchies," *IBM Systems Journal*, vol. 9, no. 2, pp. 78–117, 1970.

[24] S. Mittal, "A survey of recent prefetching techniques for processor caches," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 35, 2016.

[25] H. D. Morris, *Probability and Statistics: Classic Version*. Prentice Hall, Inc., 2018.

[26] R. C. Murphy and P. M. Kogge, "On the memory access patterns of supercomputer applications: Benchmark selection and its implications," *IEEE Transactions on Computers*, vol. 56, no. 7, pp. 937–945, 2007.

[27] M. Pavlovic, N. Puzovic, and A. Ramirez, "Data placement in HPC architectures with heterogeneous off-chip memory," in *Proc. of Int'l Conf. on Computer Design*. IEEE, 2013, pp. 193–200.

[28] G. Petrousis, "An evaluation of decoupled access execute on ARMv8," Master's thesis, Uppsala University, 2017.

[29] B. Reagen, R. Adolf, Y. S. Shao, G.-Y. Wei, and D. Brooks, "Machsuite: Benchmarks for accelerator design and customized architectures," in *Proc. of IEEE Int'l Symp. on Workload Characterization*, 2014, pp. 110–119.

[30] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[31] H. V. Simhadri, "Program-centric cost models for locality and parallelism," Ph.D. dissertation, Carnegie Mellon University, 2013.

[32] E. S. Sorenson and J. K. Flanagan, "Cache characterization surfaces and predicting workload miss rates," in *Proc. of IEEE Int'l Workshop on Workload Characterization*, 2001, pp. 129–139.

[33] ——, "Evaluating synthetic trace models using locality surfaces," in *Proc. of IEEE Int'l Workshop on Workload Characterization*, 2002, pp. 23–33.

[34] J. R. Spirn and P. J. Denning, "Experiments with program locality," in *Proc. of ACM Fall Joint Computer Conference, Part I*, ser. AFIPS, 1972, pp. 611–621.

[35] M. M. Tikir, L. Carrington, E. Strohmaier, and A. Snavely, "A genetic algorithms approach to modeling the performance of memory-bound computations," in *Proc. of ACM/IEEE Conf. on Supercomputing*, 2007.

[36] B. Verghese, S. Devine, A. Gupta, and M. Rosenblum, "Operating system support for improving data locality on CC-NUMA compute servers," *ACM SIGPLAN Notices*, vol. 31, no. 9, pp. 279–289, 1996.

[37] M. Wang, A. Ailamaki, and C. Faloutsos, "Capturing the spatio-temporal behavior of real traffic data," *Performance Evaluation*, vol. 49, no. 1-4, pp. 147–163, 2002.

[38] J. Weinberg, M. O. McCracken, E. Strohmaier, and A. Snavely, "Quantifying locality in the memory access patterns of HPC applications," in *Proc. of ACM/IEEE Conference on Supercomputing*, 2005.

[39] Y. Zhong, X. Shen, and C. Ding, "Program locality analysis using reuse distance," *ACM Transactions on Programming Languages and Systems*, vol. 31, no. 6, pp. 20:1–20:39, 2009.