

# Cache Characterization and Performance Studies Using Locality Surfaces

A Dissertation Proposal  
Presented to the  
Department of Computer Science  
Brigham Young University

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

by  
Elizabeth S. Sorenson  
February 2003

## 1 Introduction

Moore's Law states that processor speeds double every 18 months. Memory density is increasing at a similar rate, but memory speeds increase at the much slower rate of about 7% per year [1]. This means that the time needed to access memory is an increasing bottleneck. To help overcome this mismatch in speed, computer architecture designers take advantage of the fact that smaller memories placed close to the processor are significantly faster than main memory [2].

Small, fast memories between the processor and main memory are called *caches*. Caches contain a subset of the data in main memory. If a significant portion of the data the processor requires is found in the cache, the processor is relieved from waiting for slower main memory. Cache access speeds are about 1 ns and main memory access speeds are about 100 ns [1]. When the contents of an accessed memory location are in the cache, it is termed a *hit*. When the contents are not in the cache it is termed a *miss*. Large caches are more likely to have hits, but are slower and more expensive. Researchers do significant amounts of cache studies to find an appropriate trade-off.

Cache performance is frequently evaluated in terms of *miss rate*, or the number of misses for a given workload divided by the total number of memory references. Most caches are described in terms of their size (how much data the cache can hold), their line size (what size chunks of data are pulled from main memory at a given time) and their associativity (how many places in the cache a given piece of data can reside). Larger cache sizes, line sizes, and associativities tend to yield lower miss rates, but with slower access times.

Most processors today have several levels of caches. Level-one (L1) caches are next to the processor and are the smallest and fastest kind of cache. L1 caches are typically split in two pieces, one for instruction fetches and one for data reads and writes. Level-two (L2) caches are between the L1 caches and main memory. They are slightly larger and slightly slower. Higher level numbers occur if there are more layers of caches before main memory.

Caches typically store the most recently accessed data from main memory. Due to the *principle of locality*, these recently accessed parts of memory are the most likely to be re-used by the processor in the near future. Caches improve performance because most programs run on processors have large amounts of locality [3]. Locality means that after a processor accesses point  $x$  in memory,  $x$  and other locations close to  $x$  in memory tend to be accessed soon. Typically, researchers divide locality into two types. *Temporal locality* occurs when the processor reuses the same location in memory shortly after a previous use. *Spatial locality* occurs when the processor uses a close location in memory shortly after another

use [4]. Some researchers have divided locality into more types [5] [6] [7], but these two are the most common.

Many researchers have attempted to quantify locality for a variety of uses. For example, in [5] researchers used locality metrics to help modify a compiler to improve loop nest locality. In [8] locality is used to help rearrange data in memory to improve program performance. In [9] researchers used locality to study paging issues. In [10], researchers use a scalar definition of locality to help predict cache performance. [11] uses a locality metric as an input for creating synthetic traces. Grimsrud created a *locality surface* in [12] that attempted to be a general method of evaluating locality for any purpose. However, each of these locality metrics has limited application.

While many locality methods have been introduced, few are useful for evaluating caches. Cache simulation results are more often used to determine the degree of locality, rather than using locality to predict cache simulation results. In addition, no one has previously attempted to directly describe caches in terms of locality.

Most cache studies are done via trace-driven simulation. Using one method or another, a list of memory requests is recorded as a trace while a given program runs on a computer. Many methods have been proposed over the years [13] [14] [15] [16]. One of the more recent and accurate methods is BACH [17] [18]. At BYU, we have a large repository of traces collected using the BACH system. The workloads traced include SPEC CINT2000 and SPEC CFP2000 on a variety of operating systems. The traces are freely available to the public over the internet at [traces.byu.edu](http://traces.byu.edu) [19].

For trace-driven simulation, a trace is submitted to a cache simulation program for a given cache configuration. The simulator returns the miss rate, or some other metric that reflects how well the cache would have performed on the given program. This can be time consuming, especially if a large number of cache configurations are evaluated. In addition, the space necessary to store a trace of reasonable length can be quite large [14]. Other methods that either shorten the time or space necessary for this process are worth considering.

One method for reducing the storage requirements of a trace is synthetic trace generation. Rather than storing an entire trace, one must only store a limited number of parameters. From these parameters, a synthetic trace of arbitrary length can be generated. Synthetic traces were first used as a replacement for real traces, because real traces were not available. Example synthetic trace models include the Independent Reference Model [20] [21]

and the Distance Model [22]. Now, real traces are created and used to determine the necessary parameters for a model. A good synthetic trace should produce the same cache simulation results as the original real trace.

We propose improving Grimsrud’s locality surface in such a way that it will solve a number of the problems related to the performance evaluation of cache memories. Our new locality surface will help characterize workloads in terms of locality, and will give methods for qualitatively and quantitatively predicting cache performance. In addition, the locality surface is the basis for another surface that characterizes caches in terms of locality, independent of the workload. Lastly, the locality surface will help evaluate the effectiveness of previous synthetic trace generation methods and help us develop a new synthetic trace model.

## **2 Thesis Statement**

Locality surfaces provide an additional method for qualitatively and quantitatively evaluating cache performance and creating simple, more accurate synthetic traces. Locality surfaces are the basis for cache characterization surfaces that help describe how caches work in terms of locality independent of workload.

## **3 Research Description**

Without caches, modern computers would be prohibitively slow [4]. Caches are usually evaluated using trace-driven simulation. But this method is workload specific because it only tells us how a cache performs for a specific workload. We propose other methods for studying caches using cache characterization surfaces, locality surfaces, and a new method for synthetic trace creation. All of these proposed methods are based on memory reference locality, which is the defining characteristic of caches.

### **3.1 The Locality Surface**

Most researchers evaluate locality either as a scalar value [10] [11] or as a series of two-dimensional graphs [23] [24]. A scalar value oversimplifies locality. A value such as the miss rate may help one understand how a given workload would function in the given cache, but gives little indication how the same workload would function in caches of other sizes or configurations. There is no one two-dimensional graph that includes all aspects of locality, so multiple graphs are necessary. Even then, spatial locality may be evaluated in one of several ways. Some researchers find the average distance in memory between each set of successive

references to the same item. Others use information about the average length of sequential runs of references. In [25], the authors find “the probability that between references to the same item, a reference to an item  $x$  units away occurs.”

In [12] and [26], Grimsrud introduced the locality surface—a three dimensional graph that included aspects of temporal and spatial locality. It is essentially a histogram of the number of stride/delay occurrences that occur between values in a list. For any two values, the delay is the number of values between them in the list and the stride is the difference between the values. Stride can be positive or negative, but delay is only counted in one direction and is therefore always positive.

There are a few problems with the locality surface described by Grimsrud when used for the evaluation of cache memories. For example, Grimsrud’s delay is the total number of values between two given values in a list. On his locality surface, the maximum delay is therefore a function of the length of the trace. When using the locality surfaces for cache studies, Grimsrud was limited. Caches may have exactly the same number of misses, or the same miss rate, for multiple traces of different lengths. LRU (least recently used) caches are more stack based. In fact, other researchers using two-dimensional graphs used the number of unique values between two given values as a delay measure [25]. Essentially, this is a stack based approach.

We have modified Grimsrud’s locality surface to use this stack based method [27]. It is more time consuming to compute since it is an  $O(n^2)$  algorithm rather than the  $O(n)$  algorithm Grimsrud used in [12]. However, there are many advantages. The maximum delay seen on a given locality surface, which we call the *effective working set size*, indicates what size cache will comfortably contain most of the references in the trace. Once a cache has reached this size, there is little performance improvement noticed by increasing the size further.

Other features on our locality surface are evident. Figure 1 shows an example locality surface from the SPEC CINT2000 benchmark suite. Locality surfaces are typically displayed with two views making it easier to see placement and weight of different features. Temporal locality, which occurs when the same memory address appears repeatedly, can be seen along the  $stride = 0$  axis. The sequential ridge occurs where  $stride = delay$ . Sequential references occur when the processor requests succeeding locations in memory, such as accessing successive elements in an array. The height of the sequential ridge shows the amount of sequential runs in the trace. The ridge tapers off according to the distribution of the lengths.

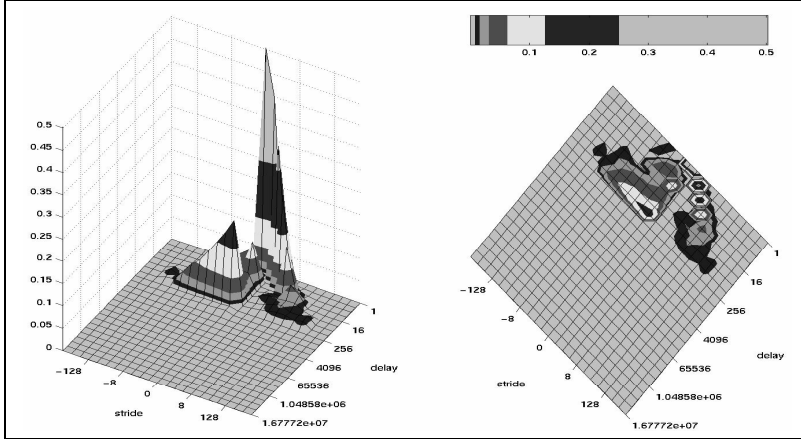


Figure 1: Example of our new locality surface. This surface is calculated from the instruction fetches of *gzip* in the SPEC CINT2000 benchmark suite.

Sharp ridges parallel to the  $delay = 1$  axis are loops. The shape of the loop gives an indication of what relationships between references are within the loop. Loop volume where the stride is negative indicates that the contents of the loop progress forward in memory. Loop volume where the stride is positive indicate jumps backward in memory within the loop. For example, in Figure 1 we see a strong loop where the delay equals 128. Because the loop mostly consists of negative strides, we know that it is primarily a forward progressing loop.

Grimsrud also presented a number of properties that are always true for his locality surface. For example, the sum of the  $delay = 1$  axis always equals one. He then used these properties to define a measure of the randomness and coverage of a given trace. These properties, such as the one just mentioned, also hold true for our locality surface. We believe that other properties exist. We can use these properties to help us understand the locality surface and to create synthetic traces, as will be mentioned in Section 3.4.

With our locality surface, we have been able to characterize workloads in a way not done before, i.e. in terms of their cache performance [27]. Workloads with similar looking locality surfaces will have similar cache performance. If the goal of a benchmark suite is to exercise caches in a variety of ways, workloads with similar surfaces would be redundant to the suite. In addition, after identifying the largest effective working set size for all benchmarks in a suite, one can tell what range of cache sizes would be adequately exercised and what cache sizes are not.

We propose using the locality surfaces we have already created for the SPEC

CINT2000 benchmark suite, and making more surfaces for the SPEC CFP2000 benchmark suite traces we have. We can then intelligently comment on what range of caches are exercised and what benchmarks are redundant in the suite in terms of cache performance. When making these locality surfaces, we separate the instruction fetches from the data reads and writes since that is the way L1 caches typically work. This effectively doubles the number of workloads, but gives a better picture for how the workloads function in L1 caches.

### 3.2 Cache Characterization Surfaces

One completely novel extension of the locality surface that we have pursued is the cache characterization surface [28]. This is an attempt to describe caches entirely based on how well various locality relationships function in the cache. Caches are typically analyzed in terms of how they perform with a specific workload. The cache characterization surface is entirely independent of the workload. Using a cache characterization surface and locality surfaces for a number of workloads, a quick look can tell one which workloads function best in the given cache. Cache characterization surfaces can also help us understand how and why caches of various configurations work.

In the past we have computed cache characterization surfaces by submitting a stream of randomly generated values to a composite locality surface program and cache simulator. The composite program keeps track of which locality relationships are associated with hits and which are associated with misses in the cache simulator. The number of misses for each stride/delay relationship are divided by the total number of values that have the given relationship. The resulting percentages are displayed on a three-dimensional graph similar to the locality surface. Where the graph is low indicates such stride/delay relationships have a low likelihood of producing misses in the given cache.

The biggest problem with creating cache characterization surfaces is the time involved. Random references have very little temporal locality and therefore result in long compute times for the locality surface. While cache characterization surfaces need only be computed once, the amount of time to compute them may be prohibitively expensive for some researchers. We propose investigating new methods for computing such surfaces.

We also propose extending the creation of cache characterization surfaces to other types of caches. So far, we have simply done standard LRU caches with a variety of sizes, line sizes, and associativities. We would like to create cache characterization surfaces for column-associative [29], skewed associative [30], and a few other kinds of caches [31]. This will hopefully give us a better insight into why these specialty caches perform better on

certain types of workloads. We will also investigate what types of caches, for example MRU (most recently used) caches, are not accurately represented by cache characterization surfaces.

### 3.3 Cache Simulation Prediction

Many researchers have attempted to use analytical models to replace cache simulations with varying degrees of success [32] [33] [34]. Most researchers simply show miss rate prediction curves versus miss rates from simulation to validate their models. In addition, the models in [33] and [34] only work for fully-associative caches. These models essentially reduce the time necessary to calculate miss rates for one kind of cache with a cost of accuracy.

Cache simulation prediction can be done for a given workload using its locality surface [28]. Qualitative prediction is simply a matter of looking at the surface. As mentioned before, the effective working set size gives an indication of the smallest cache necessary to contain most of the references made by the workload. The height and length of the sequential ridge indicates how useful a larger line size may be in the cache.

Quantitative predictions have been a little harder to do with adequate accuracy. One possible method involves taking statistics directly from the locality surface, such as the length of the sequential ridge, and using these statistics as input to a regression model.

A method we have begun pursuing involves point-wise multiplying the locality surface for the given workload with the cache characterization surface for the desired cache yielding a new surface which we term a *miss surface*. The volume of the miss surface divided by the volume of the original locality surface has some correlation with the miss rate one would get from cache simulation. However, in practice the errors can be quite large. Previous efforts have yielded errors from -2370% to 94.5% [35].

Part of the reason for this error is the fact that the volume of the locality surface does not directly correlate with the number of references. Some error may also be due to some inaccuracy in the computation of the cache characterization surface. We hope to improve the cache characterization surface creation and investigate other methods to use it and the locality surface to quantitatively predict cache performance. At the least, we propose using the cache characterization surfaces for the new types of caches to see how well this method predicts their cache performance.

### 3.4 Synthetic Traces

Historically, synthetic trace generation methods have been used for a number of reasons. Initially, there were no traces of any significant length or accuracy, so synthetic trace generation was the only way to get long, somewhat reasonable traces. Some models created at this time include the Independent Reference Model [20] [21], the Distance and Distance-Strings Models [22] [36], the Partial Markov Model [32], and the Stack Model [37]. Each of these models has a finite number of input parameters that can either be invented from thin air or pulled from a real trace.

More recently, other models have been proposed, such as the Piecewise Independent Stochastic Process Model [38] and the Random Walk Model [11]. These models use traces already taken to obtain the necessary parameters. The trace can then be discarded, and the much smaller parameters retained. Using the parameters as input, the model creates a synthetic trace with properties that are hopefully similar to the original trace.

In [39], Grimsrud used his locality surfaces to evaluate the five older traces mentioned above. We have replicated this work using our locality surfaces [40]. We have also compared the cache results of an original trace with its synthetic counterpart for each of those five older models. We plan to pursue this idea and extend it to several newer models such as the PISP model in [38] and the Random Walk Model in [11]. We will award model accuracy based on similarities between the original trace’s locality surface and the synthetic trace’s locality surface and also similar cache simulation results.

Several of the parameters necessary for these methods are similar to features on the locality surface. For example, the Distance Model uses the probability distribution of the strides that occur when references are next to each other in the trace. This is equivalent with the  $delay = 1$  axis on the locality surface. On the locality surface, however, the distribution is binned logarithmically, whereas if the distribution is computed directly for the Distance Model one can get better accuracy. If we use the values available on the locality surface, does that alter for either better or worse accuracy of the resulting synthetic trace? We propose investigating this for models with parameters on the locality surface.

We also propose presenting a new synthetic trace generation method based on the locality surface. It may involve combining several previous models and using the parameters from the locality surface. It may involve other features on the locality surface that no other models have attempted to replicate. Basically, we will try to use the locality surface to make a trace that could have generated the surface. We will analyze our new synthetic trace generation method just as rigorously as the old models.

## 4 Research Plan

We plan to create locality surfaces for a number of traces of well-accepted benchmarks. Our access to the BYU trace distribution center gives us a great advantage. These traces are accurate since they use the BACH method [17] [18] and are sufficiently long (each trace is 100 million references long). We already have locality surfaces for the instruction fetches and data reads and writes of SPEC CINT2000 under Linux, Windows NT, and Windows 2000. We plan to create surfaces for SPEC CFP2000 as well. Using these surfaces, we can characterize the workloads in terms of locality and probable memory heirarchy performance.

While making these locality surfaces, we will investigate what properties hold across all surfaces. For example, in Grimsrud's work [12] he noted that the sum along the  $delay = 1$  axis of the locality surface always equals one. This property also holds true for our locality surface. We will look for other such properties, such as how the shape of a looping structure fits with all the features at lesser delays, or if there is a cap to the total volume possible on a locality surface.

Our current method for creating cache characterization surfaces is described in Section 3.2. The random stream of references is generated by the Laplacian distribution with a scale parameter of 12,384. This distribution and parameter generates references with a wide variety of stride/delay relationships, but with preference to lower valued stride/delay combinations. One drawback to the described method is that it takes several months to create each cache characterization surface.

We will investigate new methods for creating cache characterization surfaces faster. Other disributions may produce faster results. A stopping mechanism may be able to determine at what point further computation does not change the cache characterization surface. We can also apply the knowledge gained from listing the properties that hold across all locality surfaces. If some stride/delay relationships occur infrequently, then we do not have to try to make the stream of random references have that stride/delay relationship.

We may be able to create cache characterization surfaces other ways. For example, we can determine what surface would need to be multiplied by a specific locality surface in order to yield the best miss surface (mentioned in Section 3.3) for accurate cache simulation prediction. If this multiplied surface is the same across a wide variety of locality surfaces, then it would be the desired cache characterization surface. We can compare each of the creation methods in terms of the length of time to create a surface, and in terms of accuracy, meaning how well the cache characterization surfaces yield accurate miss rate prediction.

After determining what cache characterization surface creation method is the best, we will use the algorithm to create surfaces for caches with varying size, line size, and associativity. We will also create cache characterization surfaces for column associative [29] and skewed associative [30] caches, allowing us to compare and evaluate these caches in terms of locality.

We will then use these cache characterization surfaces to determine if the cache simulation prediction method in [35] is accurate enough for use. We will be able to do this for a larger number of caches and workloads. We will also investigate other methods for effective cache simulation prediction, such as using statistics gathered only from the locality surface of a particular workload. We will perform cache simulations on a number of combinations of caches and workloads, specifically the caches for which we have cache characterization surfaces and the workloads for which we have locality surfaces. These simulation results will be used to check the effectiveness of each cache simulation prediction method.

Lastly, we will evaluate previous synthetic trace methods and propose a new synthetic trace generation technique based on the locality surface. We have already begun by implementing and evaluating five older synthetic trace methods, and comparing them with the fetches and the data reads and writes of one workload. We will also implement and include 2-3 other newer synthetic trace generation methods. We will also investigate creating the parameters for some of the synthetic traces directly from the locality surface.

We will then begin combining methods for synthetic trace generation and evaluate their effectiveness in terms of accurate locality surfaces and cache simulation results. Using the properties we found for the locality surface and the knowledge gained from examining and combining the old models, we will propose a new synthetic trace generation technique with the locality surface as its input parameter. We will evaluate the effectiveness of this technique in a similar manner to how we evaluated the other models.

To summarize, the following is a list of the tasks in this research plan:

- Create locality surfaces for a number of workloads.
- Determine what properties hold true for all locality surfaces.
- Investigate new methods for creating cache characterization surfaces.
- Create cache characterization surfaces for a variety of caches.
- Re-examine the cache simulation prediction method of [35].
- Evaluate previous synthetic trace generation methods.

- Create a new synthetic trace generation technique based on the locality surface.

#### 4.1 Artifacts

- Program for creating locality surface from a stream of numbers. Usually the number stream will be a list of address references.
- Locality surfaces for SPEC CINT2000 and SPEC CFP2000.
- Program for creating cache characterization surface.
- Cache characterization surfaces for traditional caches with various sizes, line sizes, and associativities. Also for column and skewed associativities.
- Program for generating a synthetic trace of desired length and locality given a locality surface.

#### 4.2 Delimitations

### 5 Research Papers

Already published and part of the Master's Thesis:

- Elizabeth S. Sorenson and J. Kelly Flanagan. "Using Locality Surfaces to Characterize the SpecInt 2000 Benchmark Suite." In "Workload Characterization for Emerging Computer Applications," Lizy Kurian John and Ann Marie Grizzaffi Maynard, editors, pages 101-120, Kluwer Academic Publishers, 2001.
- Elizabeth S. Sorenson and J. Kelly Flanagan. "Cache Characterization Surfaces and Predicting Workload Miss Rates." In Proceedings of the Fourth IEEE Annual Workshop on Workload Characterization, December 2001, Austin, Texas.

Published since the Master's Thesis:

- Elizabeth S. Sorenson and J. Kelly Flanagan. "Evaluating Synthetic Trace Models Using Locality Surfaces." Accepted by the Fifth IEEE Annual Workshop on Workload Characterization, November 2002, Austin, Texas.

Other proposed papers:

- A New Synthetic Trace Generation Model

- Cache Characterization Surfaces: Looking at Caches in Terms of Locality
- Locality and Cache Characterization Surfaces and How They Are Used

## 6 Contribution to Computer Science

Grimsrud’s locality surface does not help with cache studies as much as our new locality surface will. Our new locality surface will characterize workloads in terms of how they function in the memory hierarchy. Our locality surface also leads to cache characterization surfaces that characterize caches in terms of locality. Such a characterization of caches independent of the workload has never been done before to our knowledge.

Currently, cache simulation studies require multiple simulations, involving large amounts of storage and time. The locality surface offers qualitative and quantitative cache studies that reduce the time for simulations.

Current traces are hard to collect for current systems and impossible for future systems. Yet these traces are necessary for accurate simulation results. With a synthetic trace method based on the locality surface, we can create traces for arbitrary amounts and types of locality. This will reduce trace storage requirements and give researchers options for creating traces of future systems.

## 7 Possible Dissertation Outline

- Introduction: Caches and why they are important to study, usually studied with trace-driven simulation, need and motivation for each project in the dissertation
- Locality Surfaces: Grimsrud background, new changes, advantages of new surface, what different features mean, properties that always hold for all locality surfaces, workload characterization for SPEC CINT2000 and SPEC CFP2000
- Cache Characterization Surfaces: why they are useful, how computed, time involved, surfaces and interpretations for a variety of caches with varying sizes, line sizes, associativities (including direct-mapped and fully associative), and other caches such as column associative and skewed associative
- Qualitative and Quantitative Cache Result Predictions: somewhat of a replication of the Master’s thesis, include more locality surfaces (some from SPEC CFP2000) and cache simulation results, present the most accurate quantitative prediction method

found, apply to the new caches presented in previous chapter, present errors and time involved

- Analyzing Old Synthetic Trace Methods: implement methods, show necessary parameters for several traces (twolf.fet and twolf.rw and others), compare locality surfaces and cache simulation results with original trace locality surfaces, some methods include: (older) IRM, D, DS, PM, SM, and (newer) PISP Model and 1-2 other newer ones
- New Synthetic Trace Method: present a new synthetic trace method, use same traces as previous chapter and compare locality surfaces, compare cache simulation results to validate locality surface results
- Conclusions and Future Work

## 8 Dissertation Schedule

Note that several of these time frames overlap. This is because most sections of work require a large amount of computing. While I am waiting for computers to chug I can be working simultaneously on something else.

- Evaluate previous synthetic trace methods: now - Feb 2003
- Develop list of universal properties for locality surface: now - Mar 2003
- Develop and evaluate new synthetic trace generation technique: Jan 2003 - June 2003
- Finish locality surfaces and cache simulation studies for SPEC CFP2000 and SPEC CINT2000: July 2003 - Aug 2003
- Investigate new cache characterization surface ideas: Sept 2003 - Oct 2003
- Create cache characterization surfaces for a variety of caches: Nov 2003- Mar 2004
- Use new cache characterization surfaces and new locality surfaces to do more cache simulation prediction: Mar 2004 - May 2004
- Write and finalize dissertation: Apr 2004 - July 2004

## References

- [1] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, San Francisco, CA, third edition, 2003.
- [2] Alan Jay Smith. Cache memories. *Computing Surveys*, 14(3):473–530, September 1982.
- [3] P. J. Denning. The working set model for program behavior. *Communications of the ACM*, 11(5):323–333, May 1968.
- [4] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, San Mateo, CA, second edition, 1996.
- [5] Michael E. Wolf and Monica S. Lam. A data locality optimizing algorithm. In *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation*, Toronto, Ontario, Canada, June 1991.
- [6] Sangyeun Cho, Pen-Chung Yew, and Gyungho Lee. Access region locality for high-bandwidth processor memory system design. In *Proceedings of the 32nd International Symposium on Microarchitecture*, Haifa, Israel, November 1999.
- [7] Mikko H. Lipasti, Christopher B. Wilerson, and John Paul Shen. Value locality and load value prediction. In *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems VII*, October 1996.
- [8] Dan N. Truong, Francois Bodin, and Andre Sez nec. Accurate data distribution into blocks may boost cache performance. Technical Report RR-3174, 1997.
- [9] Amos Fiat and Anna R. Karlin. Randomized and multipointer paging with locality of reference. In *Proceedings of the twenty-seventh annual ACM symposium of Theory of computing*, pages 626–634. ACM Press, 1995.
- [10] Michael Salsburg. An analytical method for evaluating the performance of cache using the lru process. *Computer Measurement Group Transactions*, pages 77–87, Winter 1994.
- [11] Dominique Thiebaut, Joel L. Wolf, and Harold S. Stone. Synthetic traces for trace-driven simulation of cache memories. *IEEE Transactions on Computers*, 41(4):388–410, April 1992.
- [12] K. Grimsrud. *Visualizing Locality*. PhD thesis, Brigham Young University, 1993.

- [13] A. Agarwal, R. L. Sites, and M. Horowitz. ATUM: a new technique for capturing address traces using microcode. In *Proceedings of the 13th International Symposium on Computer Architecture*, pages 119–127. IEEE, 1986.
- [14] A. Borg, R. E. Kessler, and D. W. Wall. Generation and analysis of very long address traces. In *Proceedings of the 17th International Symposium on Computer Architecture*, pages 270–279, May 1990.
- [15] D. Nagle, R. Uhlig, and T. Mudge. Monster: A tool for analyzing the interaction between operating systems and computer architectures. Technical report, The University of Michigan, 1992.
- [16] Richard A. Uhlig and Trevor N. Mudge. Trace-driven memory simulation: A survey. *ACM Computing Surveys*, 29(2), June 1997.
- [17] J. K. Flanagan, B. Nelson, J. Archibald, and K. Grimsrud. BACH: BYU address collection hardware; the collection of complete traces. In *Proceedings of the 6th International Conference On Modeling Techniques and Tools for Computer Performance Evaluation*, September 1992.
- [18] K. Grimsrud, J. Archibald, M. Ripley, K. Flanagan, and B. Nelson. BACH: A hardware monitor for tracing microprocessor-based systems. *Microprocessors and Microsystems*, 17(6):443–459, October 1993.
- [19] Niki C. Thornock and J. Kelly Flanagan. A national trace collection and distribution resource. *ACM SIGARCH Computer Architecture News*, 29(3):6–10, 2001.
- [20] A. Aho, P. Denning, and J. Ullman. Principles of optimal page replacement. *Journal of the ACM*, pages 80–93, January 1971.
- [21] O. Aven, E. Coffman, and Y. Kogan. *Stochastic Analysis of Computer Storage*. Reidel, Amsterdam, 1987.
- [22] J. Spirn. *Program Behavior: Models and Measurements*. Elsevier North-Holland, Inc., New York, NY, 1977.
- [23] Kathryn S. McKinley and Olivier Temam. Quantifying loop nest locality using spec’95 and the perfect benchmarks. *ACM Transactions on Computer Systems*, 17(4):288–336, November 1999.

- [24] F. J. Sanchez and A. Gonzalez. Data locality analysis of the specfp95. *Digest of Performance Analysis and its Impact on Design (PAID) Workshop*, pages 78–84, 1998.
- [25] Thomas M. Conte and Wen mei W. Hwu. Benchmark characterization for experimental system evaluation. In *Proceedings of the 1990 Hawaii International Conference on System Sciences (HICSS)*, volume I of *Architecture Track*, pages 6–18, 1990.
- [26] K. Grimsrud, J. Archibald, R. Frost, and B. Nelson. Locality as a visualization tool. *IEEE Transactions On Computers*, 45(11), November 1996.
- [27] Elizabeth S. Sorenson and J. Kelly Flanagan. Using locality surfaces to characterize the specint 2000 benchmark suite. In Lizy Kurian John and Ann Marie Grizzaffi Maynard, editors, *Workload Characterization of Emerging Computer Applications*, pages 101–120. Kluwer Academic Publishers, 2001.
- [28] Elizabeth S. Sorenson and J. Kelly Flanagan. Cache characterization surfaces and prediction workload miss rates. In *Proceedings of the Fourth IEEE Annual Workshop on Workload Characterization*, pages 129–139, December 2001.
- [29] Anant Agarwal and Steven D. Pudar. Column-associative caches: A technique for reducing the miss rate of direct-mapped caches. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 179–190. ACM Press, 1993.
- [30] Andre Seznec. A case for two-way skewed-associative caches. In *Proceedings of the 20th Annual Symposium on Computer Architecture*, pages 169–178, May 1993.
- [31] Toni Juan, Tomas Lang, and Juan J. Navarro. The difference-bit cache. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 114–120, May 1996.
- [32] A. Agarwal, M. Horowitz, and J. Hennessy. An analytical cache model. *ACM Transactions on Computer Systems*, 7(2):184–215, May 1989.
- [33] Jaswinder Pal Singh, Harold S. Stone, and Dominique F. Thiebaut. A model of workloads and its use in miss-rate prediction for fully associative caches. *IEEE Transactions on Computers*, 41(7):811–825, July 1992.
- [34] Che-Chi Weng and Eric E. Johnson. The time-space model for instruction reference behavior. In *Proceedings of the 1994 IEEE International Phoenix Conference on Computers and Communications*, pages 220–226, April 1994.

- [35] Elizabeth S. Sorenson. Using locality to predict cache performance. Master's thesis, Brigham Young University, 2001.
- [36] C. Fricker and P. Robert. A memory reference model for the analysis of cache memories. *Performance '90*, pages 255–269, 1990.
- [37] J. Archibald and J-L. Baer. Cache coherence protocols: Evaluation using a multi-processor simulation model. *ACM Transactions on Computer Systems*, 4(4):273–298, November 1986.
- [38] Anup Mathur. *A Stochastic Process Model for Transient Trace Data*. PhD thesis, Virginia Polytechnic Institute and State University, 1996.
- [39] K. Grimsrud, J. Archibald, R. Frost, and B. Nelson. On the accuracy of memory reference models. In *7th International Conference Proceedings*, pages 369–388, Springer-Verlag, May 1994.
- [40] Elizabeth S. Sorenson and J. Kelly Flangan. Evaluating synthetic trace models using locality surfaces. In *Proceedings of the Fifth IEEE Annual Workshop on Workload Characterization*, pages 23–33, November 2002.

This dissertation proposal by Elizabeth S. Sorenson is accepted in its present form by the Department of Computer Science of Brigham Young University as satisfying the dissertation proposal requirement for the degree of Doctor of Philosophy.

---

J. Kelly Flanagan, Committee Chair

---

Michael Goodrich, Committee Member

---

Bryan Morse, Committee Member

---

Dan R. Olsen, Committee Member

---

William A. Barrett, Committee Member

---

David W. Embley, Graduate Coordinator