Visualizing I/O Predictability

Alison Luo, Ahmed Amer, Scott Speidel, Darrell D. E. Long, Alex Pang
Jack Baskin School of Engineering
University of California, Santa Cruz
{*alison,amer4,rss,darrell,pang*}@*soe.ucsc.edu*

Abstract

We propose a novel method to study storage system predictability based on the visualization of file successor entropy, a form of conditional entropy drawn from a file access trace. First-order conditional entropy can be used as a measure o f predictability. It is superior to the more common measures such as independent likelihood of data access. For file access data, we developed a visualization tool that produces 3D graphical views of the variation in predictability of successive access events on a per-file basis. Our visualization tool provides interactive observation of the variations in predictability according to some arbitrary criterion, e.g. time of day, program identifier, user groups, or any other classification of files. Four entropy data sets were extracted from various file system traces. These four data sets are representative of the variability in file access patterns for different machine use: server, personal workstation, large number of interactive users, and heavy write activity. Visualization results show that there is strong predictability among files and optimizations would be profitable.

Key Words and Phrases: conditional entropy, file access trace, data access patterns, caching effects

1. Introduction

I/O and storage systems are a major performance bottleneck [15], and this situation is growing more critical as processor and bandwidth performance out-paces advances in storage access speeds. In other words, with current processor and storage performance trends, access latency to storage is a growing problem. Predictive techniques have been applied to improve the performance of I/O systems, and are an effective method of limiting the effects of this growing performance gap. The viability of such optimization depends heavily on the predictability of the specific workload. It is therefore important to study the predictability of file access patterns [10]. The usefulness or effectiveness of predictive prefetching and of other efforts at predicting file access events would be encouraged for a particular workload if there is an efficient mechanism for determining the predictability of a workload.

This study focuses on how to visualize the inherent predictability of file access sequences. Visualization of file access predictability is highly desirable in itself, as it is generally difficult to demonstrate the dependency among files given a large file system trace (*i.e.* a lengthy recording of real world file access events), and such visualization is of direct benefit to understanding the nature and structure of file access events. Such visualization is also beneficial in evaluating the effect of different parameters on predictability. For this purpose we have developed a system [12] to study the variation of file access predictability across all files for some arbitrary variable parameter. The predictability of the traces is presented in terms of conditional entropy. That is, given that a file was just accessed, how uncertain are you about which file will be accessed next. For the purpose of this work, we present visualizations of data from experiments to study the temporal variation of file access entropy over different observation periods (trace durations), as well as results for experiments that evaluate the effects of caches on modifying the predictability of workloads.

File and data accesses are driven by applications and are known to be highly predictable, and yet the problem of modeling a data access trace is far from solved [7]. Numerous studies attempt to characterize and describe different file and data access workloads [18, 19, 16, 17, 3, 14, 9], but no consistent mechanism exists to quickly and visually evaluate the nature of a workload across all files. Our system is a novel approach to evaluating file system traces, and has demonstrated considerable usefulness for file systems research. Specifically, we allow the evaluation of a workload across all files and using different variable criteria, and are capable of presenting two file and variable-specific datums simultaneously. We introduce some background concepts of entropy and how we calculate successor entropy from file traces in Section 2. The visualization system itself is described in Section 3. Experiments with traces of file accesses extending up to a year were conducted, and the resulting visualizations presented new insights into the long term nature of file predictability. These results are discussed in Section 4.

2. Entropy and Self-Information

Self-information, entropy and contextual predictability are aspects of storage systems research that have appeared very recently, most often under the context of data access prediction for prefetching [11, 10, 21, 9, 8]. Entropy is simply a measure of "disorder," first proposed in 1865 by Rudolf Clausius [5]. The concept was first described in the context of chemistry and physics, specifically thermodynamics. It was later defined in an abstract mathematical sense by Claude Shannon [20], in a work that helped found the field of information theory.

It is often the case that a file's context (relative position) in an access sequence is far more important than its frequency of occurrence. Prior work on storage system optimization, such as that of C. K. Wong [22],is heavily based on an assumption of independent probability of access. It was not until recent work by Krishnan *et al.* [21, 6] that a more accurate look at file access probabilities was considered. Even more recent work on optimal placement for tertiary storage [4], is also based on analytical arguments that assume independent probability of access for each data location.

2.1. Predictability & File Successor Entropy

Assuming a data source with an alphabet of m symbols, the self-information, H, of a sequence of symbols, S, is often calculated as follows:

$$H \quad = \quad -\sum_{i=1}^{m} \frac{n_i}{N} \cdot log(\frac{n_i}{N}) \qquad (1)$$

The value n_i is simply the frequency of occurrence of the i^{th} symbol, s_i, of the alphabet in the sequence S, and N is simply the sum of all such frequencies, *i.e.* the length of the sequence. A more precise definition of the entropy of the sequence S is based on the probability of occurrence of symbol s_i, $p_i = P(s_i)$. In fact, the value $\frac{n_i}{N}$ is simply an approximation of p_i.

It is possible in many scenarios (and file accesses are actually an excellent example of this), that p_i and $\frac{n_i}{N}$ are very different quantities. In particular, if we have somehow acquired some knowledge about the sequence, resulting in our being in a state of knowledge B, a more accurate measure to apply would be the conditional entropy:

$$H \quad = \quad -\sum_{i=1}^{m} P(s_i|B) \cdot log(P(s_i|B)) \qquad (2)$$

Where $P(s_i|B)$ is simply the probability of occurrence of symbol s_i given the knowledge that we are in state B. A definition of such a state B, and other features specific to a file access sequence are the subject of the following section.

In this work, we have examined four sets of file access traces. Each trace consists of a sequence of files that were accessed. For each file f, all recorded successors of f are candidates for a prediction. If file f is used as the condition in Equation 2, then we have $H(f)$ as a measure of the amount of disorder among its immediate successors. The higher this value, the less promising the possibility of an accurate successor prediction. Simply plotting the values of $H(f)$ for each file encountered in a trace would provide a histogram that gives a good idea of the variation in conditional entropy over the file space.

2.2. Data Sets

File access data was drawn from file system traces gathered using Carnegie Mellon University's DFSTrace system [13]. These are actual recording of all file access requests made on the systems being monitored, and the trace period used covers more than a year of system activity. The data sets used cover five systems for durations ranging from a single day to over a year. The traces represent varied workloads, particularly *mozart* a personal workstation, *ives*, a system with the largest number of users, *dvorak* a system with the largest proportion of write activity, and *barber* a server with the highest number of system calls per second.

3. The Visualization System

Input to our system consists of file entropy data calculated from the file traces. Each conditional entropy value is related to a file during a certain trace period. These data are presented in 3D by mapping the file identifiers, sorted by their conditional entropy, to the x-axis; the different durations to the y-axis; and the conditional entropy value to the z-axis and possibly color. Color defaults to mirror the z-axis information, but has also been used to represent a fourth dimension – file access frequencies. For the results we present in §4, we study both the effect of different trace durations and of different caches on the successor predictability of our file access workloads.

1. Varying time scales – each point on the surface represents the successor entropy (in both height and color), of a particular file (on x), for a particular trace duration (varied on the y-axis).

2. Varying cache sizes – each point on the surface again represents the successor entropy (in height) and the popularity (in color), of a particular file (on x), for a particular intervening cache size (varied on the y-axis).

3.1 Varying Time Scales

Because we want to study the variability of file access patterns for different durations, we keep a separate record of the access patterns for each duration. For example, over a one day period with 20,000 file access events, if we want a granularity of 5,000 events, then we have 4 distributions of conditional entropies consisting of the: first 5,000 events, first 10,000 events, first 15,000 events, and all 20,000 events. Note that because there is a different number of files accessed for each of these string of events, and because we are sorting the file identifiers on x by their conditional entropy values, points with the same x coordinate do not refer to the same file in general. For this reason it is important to point out that points along the x-axis are ranked by their conditional entropy, and not by their file identifiers. This ensures an easy mechanism for visualizing the variation of predictability among files in general, and not be limited to file identifications. Furthermore, we also want to compare the patterns of different durations. Hence, we provide two views of the same data: unnormalized (*e.g.*, Figure 1(a)) and normalized (*e.g.*, Figure 1(b)). The normalized view linearly interpolates and scales the distributions of different durations so that they all appear to be of the same length.

The conditional entropy values are mapped to both height and color. Whether they are presented as a surface (*e.g.*, Figure 1) or as a set of curves. we use a "temperature" color map where "hotter" colors are used for high entropy values (low predictability) , "cooler" colors are used for low values (high predictability).

Algorithm 1 Calculate RGB value for graph point

```
$C_{ni} \leftarrow \frac{C_i - C_{min}}{C_{max} - C_{min}}$
$R_i \leftarrow C_{ni}$
$B_i \leftarrow 1 - Cni$
if $H_{ni} \leq 0.5$ then
    $G_i \leftarrow R_i$
else
    $G_i \leftarrow B_i$
end if
```

Each data point is rendered in *RGB*, as described by Algorithm 1, either based on its entropy value or based on a fourth dimension of data if available, C_i is the value at this point for which we wish to use color information. C_{min} and C_{max} are the minimum and maximum entropy of the data set. C_{ni} is the normalized value, which ranges from 0 to 1. R_i, B_i and G_i are the RGB assignments for this point. Therefore a point datum is represented with colors, high values appear bright and red while lower values appear as dark blue. We used this color information to distinguish the frequency of file accesses performed for each file when we evaluate the effects of caching on access predictability. This is a particularly suitable choice, as the data represents a fourth dimension of data points that vary widely, avoiding any issues with the rainbow color scheme.

Because the number of files accessed can be very large (over a million), in our visualization system, all three dimensions of data can be normalized, while still preserving the property of relative file accesses predictability for effective evaluation. Finally, our system provides interactive manipulation of the data set in $3D$ space to help the user gain a better understanding of the visualized entropy.

3.2 Cache Effects & Popularity

Our second set of experiments looked at the effects of intervening caches, and file popularity. When a file access sequence is passed through a cache, we get a new "filtered" workload, representing the original data requests with the exception of those that could be satisfied by the cache. Such a filtered workload is normally considered more difficult to manage, as it often results in poor cache performance (when the filtered workload is presented to another cache, an increasingly common occurrence with today's environments). To study the effect of such filtering, we used our system to visualize file successor predictability using intervening cache size as the optional parameter (instead of trace duration, used in the time-scale experiments).

As with the time-scale experiments, each data point is rendered in *RGB* according to Algorithm 1, but the color is used to provide an additional dimension of information. Specifically, we use the color to represent the popularity of the files – the total number of times they are accessed. This allows us to visually evaluate whether there is any correlation between file access frequency and predictability. As we shall see in §4, any such correlation is limited and totally masked by increased caching.

4. Results and Discussion

In this section we will present sample results and analysis from our experiments. We start with the experiments that dealt with the effects of time scale on predictability, which also includes a presentation of the usefulness of normalization for comparing unequal histograms. We then go on to present results for the effects of different cache sizes on predictability. This also includes the use of color to demonstrate the lack of correlation between file access frequency and predictability.

4.1 Multiple Time Scales

To study the effect of time scale, the conditional entropy of file accesses are calculated. The distributions of the conditional entropy after every 100 thousand accesses are recorded cumulatively, thereby producing a histogram for the first 100 thousand accesses, then a histogram for the first 200 thousand accesses, and so on. A one year trace could thus divided into 30 histograms. We also used one day file traces from the same systems, each one with around 20 thousand file accesses observed. For the one day file traces, histograms of conditional entropy are produced after every five thousand file access events.

Figure 1 shows entropy data extracted from a one day trace of the workstation *mozart*. Since there are 20,000 access events per day and we are studying the traces in increments of 5,000 access events, there are 4 curves representing durations of 5,000, 10,000, 15,000 and 20,000 file access events. Each curve represents conditional entropy calculated for a different fraction of the day long trace. These are paired and tiled to generate the surface in Figure 1. Because this implies a different number of file identifiers at each increment (Figure 1(a)), we have the option to normalize file identifier numbers (along the *x* axis) to an adjustable consistent range (Figure 1(b)). This allows us to evaluate variations in the distribution of files with high and low entropy, without surface variation introduced by variations in the number of files. It is clear that the entropy curves decrease very fast and remain at low values for most of the day.

More interesting observations can be made from a look at the longer trace durations, which range up to a full year (Figure 2). The most interesting observation from Figure 2 is the general similarity among the workloads, and their dissimilarity with the single day trace of *mozart*. This similarity true for all four workloads tested, though the figure shows only *mozart* and *barber*. The entropy surfaces of all four systems are very similar and most areas are blue. This shows that the entropy stays at very low values and the predictability doesn't change much from system to system. So we can conclude that successor predictability is largely system independent. The second important observation is the reduced variation in a longer trace period (one year of Figure 2(a) *vs.* one day of Figure 1(b)). This is a very strong suggestion that file system predictability is very high in general, but that file prefetching performance would be improved with longer trace durations. Longer trace periods would allow more detailed conclusions to be drawn, as more successor information can be gathered.

The even nature of this effect and the extended durations over which it holds true, have been very informative for systems researchers using these traces. One particular work on this subject involved the construction of a predictive cache which utilized sets of successors of variable size. Larger sets seemed to be of limited use for short multi-day simulations. Simulations of these larger sets over periods of almost a year resulted in dramatic performance gains [2]. Knowledge of the behavior of the workload over extended durations was directly responsible for demonstrating an almost 60% reduction in requests for out-of-cache files.

4.2 Caching Effects

For caching effects, we used our system to visualize the effects of increasing cache sizes on the workload. This was done by filtering the workloads through a simulated cache of capacity ranging from 0 to 300, and plotting the conditional file entropy histograms for each cache capacity. A capacity of 0 is equivalent to the original access sequence, while a cache of size 300 represents a large enough cache capacity to render simple independent probability based predictions useless.

We present the results of these experiments for access sequences of approximately a month (Figure 3). Similar results were observed for longer durations. The most important observation to be made from these figures is how the most popular files are widely distributed across the range of predictable files. Specifically, aside from the original workloads (with cache size 0), there is limited correlation between predictability and access frequency (we see light points across the range of files). This is especially important as it clearly demonstrates that many popular files are also very predictable in their associated access behavior. These observations support prior work that has demonstrated how a succession-based predictive caching scheme can maintain good performance in spite of intervening caches [1].

5. Conclusion and Future Work

We have presented a novel technique to view the inherent relatedness of arbitrary file access sequences, and its variability with respect to an arbitrary parameter. For this case study, we looked at temporal variability, and the effects of intervening caches of varying sizes. Specifically, we used groups of file successors and their conditional entropy. We then used either increasing trace durations, or increasing cache sizes as the parameter. The resulting 3D entropy curves and surfaces demonstrated that subsequent file accesses are highly predictable, and that variation in this result diminishes as the duration of the observation period increases. These observations were of direct benefit to a current systems research project that shows promising results in improving file caching performance. In general being able to identify workloads with such a high dependency among file accesses suggests great promise for the application of predictive optimizations to those workloads.

We were also able to demonstrate that some of the most frequently accessed files are also among the most predictable, an observation that remained undiminished by intervening caches.

For the entropy data, we need to view data generated with even more parametric conditions (mapped to *y*-axis). It would be interesting to study entropy of user-classified file access events, or program-classified access events. For the time being we have only considered time variation, and caching effects. Yet in doing so, we have been able to provide strong visual evidence for the consistently predictable file accesses over widely varying time scales.

It is also important to demonstrate the generality of this approach by testing, using different conditioning predicates (the *x*-axis entries). For now we have limited these to single file identifiers for the predicating condition. Candidates for predicating conditions include file groups and sequences, accesses filtered through a cache, and physical storage blocks of varying sizes. Other possibilities for parametric conditions (*y*-axis definitions) include different count-decay policies, varying block, group, or sequence sizes, and even competing predicating policies (*e.g.* context modeling *vs* pattern detection *vs* program-based modeling).

6. Acknowledgment

The authors would like to thank support from the National Science Foundation award CCR-9972212, the Usenix Association, and NSF ACI-9908881.

References

[1] A. Amer and D. D. E. Long. Adverse filtering effects and the resilience of aggregating caches. In *Proceedings of the Workshop on Caching, Coherence and Consistency (WC3 '01)*, Sorrento, Italy, June 2001. ACM.

[2] A. Amer and D. D. E. Long. Aggregating caches: A mechanism for implicit file prefetching. In *Proceedings of the Ninth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2001)*, pages 293–301, Cincinnati, OH, Aug. 2001. IEEE.

[3] J. Choi, S. H. Noh, S. L. Min, and Y. Cho. Towards application/file-level characterization of block references: A case for fine-grained buffer management. In *Proceeding of ACM SIGMETRICS 2000*, Santa Clara, CA, June 2000.

[4] S. Christodoulakis, P. Triantafillou, and F. A. Zioga. Principles of optimally placing data in tertiary storage libraries. In *Proceedings of the 23rd VLDB Conference*, Athens, Greece, 1997.

[5] R. Clausius. Ueber verschiedene für die Anwendung bequeme Formen der Hauptgleichungen der mechanischen Wärmetheorie. *Ann. Phys. Chemie*, 125:353–400, 1865.

[6] K. M. Curewitz, P. Krishnan, and J. S. Vitter. Practical prefetching via data compression. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93)*, pages 257–266, Washington, D. C., May 1993.

[7] G. R. Ganger. Generating representative synthetic workloads: An unsolved problem. In *Proceedings of the Computer Measurement Group (CMG) Conference*, pages 263–1269, Dec. 1995.

[8] P. Krishnan. *Online Prediction Algorithms for Databases and Operating Systems*. PhD thesis, Department of Computer Science, Brown University, Providence, Rhode Island 02912, Aug. 1995.

[9] T. M. Kroeger. *Modeling File Access Patterns to Improve Caching Performance*. PhD thesis, University of California, Santa Cruz, Mar. 2000.

[10] T. M. Kroeger and D. D. E. Long. The case for efficient file access pattern modeling. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems (HotOS-VII)*, Rio Rico, Arizona, Mar. 1999. IEEE.

[11] T. M. Kroeger and D. D. E. Long. Design and implementation of a predictive file prefetching algorithm. In *Proceedings of the 2001 USENIX Annual Technical Conference*, Boston, MA, June 2001.

[12] A. Luo, A. Amer, N. Der, D. D. E. Long, and A. Pang. Visualizing file system predictability. In *Works In Progress at IEEE Visualization 2001*, San Diego, CA, Oct. 2001. IEEE.

[13] L. Mummert and M. Satyanarayanan. Long term distributed file reference tracing: Implementation and experience. *Software - Practice and Experience (SPE)*, 26(6):705–736, June 1996.

[14] N. Nieuwejaar, D. Kotz, A. Purakayastha, C. S. Ellis, and M. Best. File-access characteristics of parallel scientific workloads. *IEEE Transactions on Parallel and Distributed Systems*, 7(10):1075–1089, October 1996.

[15] J. K. Ousterhout. Why aren't operating systems getting faster as fast as hardware? In *USENIX Summer Conference*, Anaheim, California, June 1990.

[16] D. Roselli. Characteristics of file system workloads. Technical Report CSD-98-1029, University of California, Berkeley, Dec. 1998.

[17] D. Roselli, J. R. Lorch, and T. E. Anderson. A comparison of file system workloads. In *Proceedings of the 2000 USENIX Annual Technical Conferenc*, San Diego, CA, June 2000.

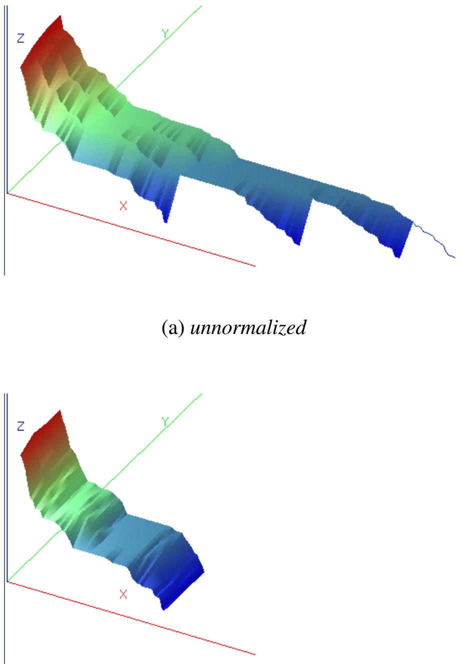
[18] C. Ruemmler and J. Wilkes. A trace-driven analysis of disk working-set sizes. Technical Report HPL-OSR-93-23, Hewlett-Packard Laboratories, Palo Alto, CA, Apr. 1991.

[19] C. Ruemmler and J. Wilkes. UNIX disk access patterns. In *Proceedings of the Usenix Technical Conference*, pages 405–420, San Diego, CA, Winter 1993. Usenix Association.

[20] C. E. Shannon. *The mathematical theory of communication*. University of Illinois Press, Urbana, 1 edition, 1949.

[21] J. S. Vitter and P. Krishnan. Optimal prefetching via data compression. *Journal of the ACM*, 43(5):771–93, Sept. 1996.

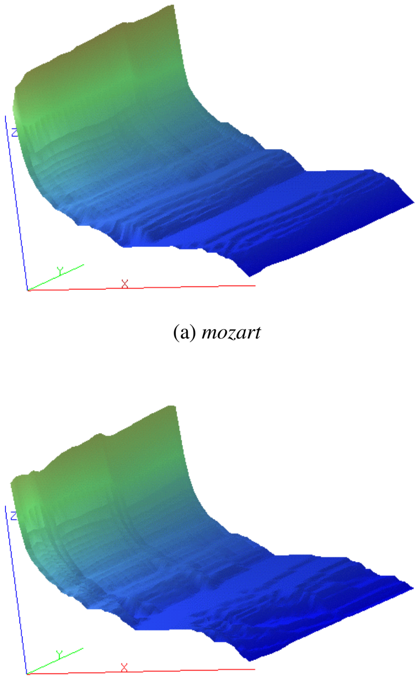
[22] C. K. Wong. Minimizing expected head movement in one-dimensional and two-dimensional mass storage systems. *Computing Surveys*, 12(2):167–177, June 1980.



(a) *unnormalized*

(b) *normalized*

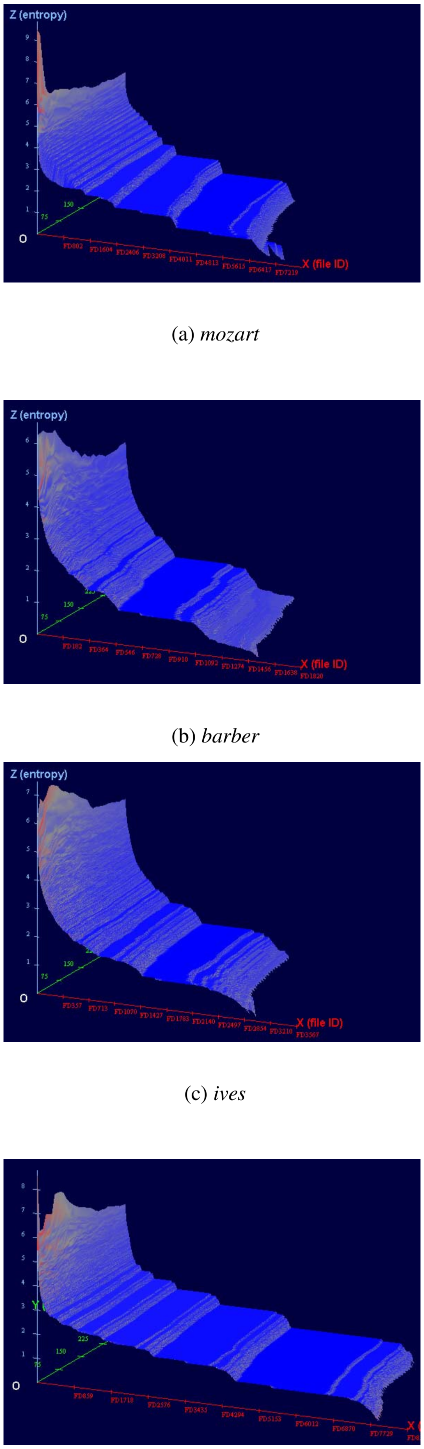
Figure 1. One day data from *mozart*



(a) *mozart*

(b) *barber*

Figure 2. Varying Time Scale Surface Visualization



(a) *mozart*

(b) *barber*

(c) *ives*

(d) *dvorak*

Figure 3. Cache-Frequency Visualization – Month