# Scalar Metric for Temporal Locality and Estimation of Cache Performance

Juha Alakarhu and Jarkko Niittylahti
Institute of Digital and Computer Systems, Tampere University of Technology
P.O. Box 553, FIN-33101 Tampere, Finland
juha.alakarhu@tut.fi, jarkko.niittylahti@tut.fi

## Abstract

*A scalar metric for temporal locality is proposed. The metric is based on LRU stack distance. This paper shows that the cache hit rate can be estimated based on the proposed metric (an error of a few percents can be expected). The metric alleviates high-level memory system outlining and enables using stack processing in run-time locality analysis.*

## 1. Introduction

The performance of a memory hierarchy heavily depends on the locality of the memory references. A well-defined metric for locality would alleviate designing and studying memory systems. Traditionally, locality has been characterized verbally (“a lot” or “little”) or visually using stack distance curves [3]. There has not been a commonly accepted scalar metric. This paper presents such metric for temporal locality.

The proposed metric is based on (LRU) stack distance curve, which is considered an excellent metric of temporal locality, e.g., [1]. A scalar metric, instead of a curve, alleviates comparing and distributing locality information. It also enables using stack processing techniques for studying run-time behavior of a program. The lack of run-time information has been earlier considered as a drawback of studies using stack processing [5].

Originally, stack processing was proposed to be a one-pass simulation technique for all cache sizes [3]. An LRU stack distance curve states the probability for there being $x$ or less unique addresses between successive references to the same address. The hit rate of a fully associative cache is directly shown by the curve, and the hit rate of a set associative curve can be fairly accurately estimated [4].

## 2. Scalar metric for temporal locality

Temporal locality has been traditionally studied by visually exploring stack distance curves using a logarithmic scale for $x$. A curve that is higher in such diagram is considered to have more temporal locality.

The metric proposed in this paper is calculated as follows

$$T(F^T) = \sum_{i=0}^{\log_2 x_m} \frac{F^T(2^i)}{\log_2 x_m + 1}, \qquad (1)$$

i.e., the stack distance curve $F^T(x)$ is averaged at x’s values $1, 2, 4, \ldots, x_m$, evaluating to a value between zero and one. The result is larger for a curve that would be considered to have more temporal locality by visually studying the curves.

The metric is demonstrated in Fig. 1 and in the header of Fig. 2. They show the run-time behavior of two test applications and the locality values for the pre- and post- L1 accesses for a direct mapped and fully associative cache.

## 3. Regenerating stack distance curve

A simple method for regenerating the original curve based on the scalar value $L_t$ is presented. The product of the following functions is used to outline the curve: $f^A(x) = 1 - \beta x^{\alpha}$, $(\alpha < 0)$, $f^B(x) = 1 - \delta^{\frac{x}{\gamma}}$, $f^C(x) = \min((\frac{x}{\varepsilon})^{\zeta}, 1)$. $f^A(x)$ is based on a rule of thumb for increasing the cache size [2], $f^B(x)$ the way a fully associative cache filters memory stream (no short stack distances in the output), and $f^C(x)$ the way a low-associativity cache filters the stream (there are also short stack distances due to conflict misses).

In the following, the constants $\alpha - \zeta$ are set without knowing the source of the memory sequence. First, set $\beta = 1 - 0.7L_t$ and find $\alpha$ such that $T(f^A(x)) = \max(L_t, A)$. Setting $\beta$ approximates the starting point of $f^A(x)$ to be $0.7L_t$. Then, set $\zeta = 0.3$ and find $\varepsilon$, such that $T(f^A(x)f^C(x)) =$

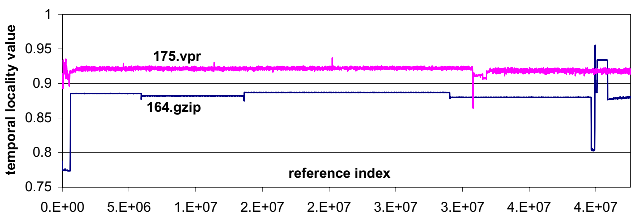

**Figure 1. Temporal locality and ref. index.**

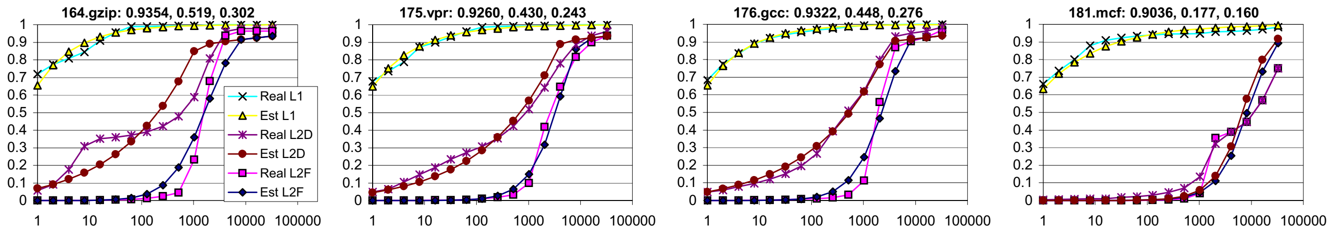

**Figure 2. Scalar values and actual and estimated curves (pre-L1, post-L1 direct and fully assoc.).**

$\max(L_t, C)$. The typical curvature of the curve after a low-associativity cache is approximated by the value 0.3. Highly associative caches are also handled by $f^B(x)$. Finally, set $\delta = 0.7$ and find $\gamma$, such that $T\left(f^A(x)\, f^B(x)\, f^C(x)\right) = L_t$. The value of $\delta(<1, >0)$ does not affect the form of the curve.

The source of the sequence being unknown, we have to provide limits for translating from one form to another ($A = 0.8$ and $C = 0.4$). The values are not critical as the form of the estimated function changes smoothly. Usually, the source of the memory sequence does not have to be concluded from the scalar value, and the correct functions can be applied directly.

We do not claim this method to be the best for regenerating the curve. Rather than emphasizing the method itself, we point out that it *is* possible to estimate the curve based on the proposed scalar metric and certain principles.

## 4. Cache performance estimation

To test the metric in cache performance estimation, simulations were run using SPECint 2000 benchmarks. The references in the input and output of a direct mapped and a fully associative (64 KB / 64 B) L1 cache were studied.

Fig. 2 compares the actual curves to the estimates obtained using the method presented in the previous section for the first four benchmarks. Including all 12 benchmarks, the absolute average and worst-case deviations are 1.3 % / 6.4 %, 6.7 % / 26 % and 4.8 % / 28 % for pre-L1, post-L1 direct mapped associative, and post-L1 fully associative, respectively.

A single value of the curve directly states the hit rate of a fully associative cache. However, for set-associative systems, the full curve is used, as follows [4]

$$h(n, s, f^T) = \sum_{i=1}^{n} \sum_{j=i}^{\infty} \binom{j-1}{i-1} \left[\frac{1}{s}\right]^{i-1} \left[\frac{s-1}{s}\right]^{j-i} f^T(j), \quad (2)$$

where $n$ states the degree of associativity, $s$ the number of sets, and $f^T$ is the derivate of the estimated curve. Thus, because $T(F_e(x)) = T(F^T(x))$, local deviations are filtered away and a more accurate estimate is obtained for set-associative than for fully associative cache.

Table 1 shows the cache hit rate estimates for direct mapped caches: 16 KB / 64 B L1 and 512 KB / 64 B L2, backing a direct mapped (L2-A) and fully associative (L2-B) L1. Here, the average and worst-case errors including all the benchmarks are 0.58 % / 1.4 % (L1), 5.1 % / 9.4 % (L2-A), and 4.1 % / 7.1 % (L2-B).

**Table 1. Direct mapped cache hit rates.**

| | L1 | | L2-A | | L2-B | |
|---|---|---|---|---|---|---|
| App. | Real | Est. | Real | Est. | Real | Est. |
| 164.gzip | 0.973 | 0.968 | 0.84 | 0.86 | 0.72 | 0.68 |
| 175.vpr | 0.963 | 0.957 | 0.73 | 0.78 | 0.55 | 0.54 |
| 176.gcc | 0.958 | 0.965 | 0.82 | 0.80 | 0.66 | 0.63 |
| 181.mcf | 0.930 | 0.930 | 0.35 | 0.35 | 0.34 | 0.30 |

## 5. Conclusions

A scalar metric for temporal locality has been proposed. The metric is more accurate than verbally describing locality, and it is easier to compare and distribute than stack distance curves. In addition, the metric enables using stack processing in run-time locality analysis. According to the results, the cache performance can be estimated with an absolute error of a few percents, even though the complete temporal behavior of a program is captured into single scalar value. This shows that the metric correlates to the cache performance, making it practically usable. Yet, the metric is no substitute for cache simulations. Rather than using the metric for obtaining cache hit rates, typical applications can benefit from comparing the metric and observing its changes.

## References

[1] M. Brehob and R. Enbody. An analytical model of locality and caching. Technical report, Michigan State University, August 1999. MSU-CSE-99-31.

[2] B. L. Jacob, P. M. Chen, S. R. Silverman, and T. N. Mudge. An analytical model for designing memory hierarchies. *IEEE Trans. Computers*, pages 1180–1194, October 1996.

[3] R. Mattson, J. Gecsei, D. Slutz, and I. Traiger. Evaluation techniques for storage hierarchies. *IBM Systems Journal*, 12(2):78–117, 1970.

[4] A. J. Smith. A comparative study of set associative memory mapping algorithms and their use for cache and main memory. *IEEE Trans. on Software Engineering*, 4(1):121–130, March 1978.

[5] D. A. B. Weikle, S. A. McKee, and W. A. Wulf. Caches as filters: a new approach to cache analysis. In *Proc. 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecom Systems*, pages 2–12, Montreal, Que., Canada, July 1998.