# PAGING PERFORMANCE WITH PAGE COLORING

**William L. Lynch and M. J. Flynn**

**Technical Report CSL-TR-91-492**

**October 1991**

The work described herein was supported in part by NASA under contract NAG2-248 using facilities supplied under NAG W 419.

# Paging Performance with Page Coloring

by

William L. Lynch and M. J. Flynn

**Technical Report CSL-TR-91-492**

October 199 1

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-4055

**Abstract**

Constraining the mapping of virtual to physical addresses (page coloring) can speed and/or simplify caches in the presence of virtual memory. For the mapping to hold, physical memory must be partitioned into distinct colors, and virtual pages allocated to a specific color of physical page determined by the mapping. This paper uses and analytical model and simulation to compare the paging effects of colored versus uncolored (conventional) page allocation, and concludes that these effects are small.

**Key Words and Phrases:** Virtual Memory, Paging, Cache Memory

Copyright © 1991
by
William L. Lynch and M. J. Flynn

# Contents

# 1 Introduction

Many computers use virtual memory to provide large address spaces for executing processes. Virtual memory requires a mapping from the virtual address space of the software to the physical address space of the hardware. A page fault occurs when the operating system must allocate a physical page in which to store the virtual page.

Conventional page allocation typically approximates least-recently-used (LRU) page replacement. The methods of approximation share the common feature of a free *list* of available pages[PS85], typically maintained as FIFO queues. Normally, there is no correlation maintained between a requested allocation from virtual memory and its placement in physical memory, as whatever physical page is on top of the free list is used. However, page coloring [BLF90] requires a very specific mapping between virtual and physical addresses to improve cache performance and design. This study examines the effects of this mapping requirement.

This study assumes a many-to-one function page-coloring methodology, which can be implemented by multiple free lists (one per color). When the operating system receives a request for a mapping for a virtual address, it removes a properly-colored physical page off that color free-list. Similarly, the operating systems adds freed pages to the proper-color free list. Thus, the difference between uncolored and colored paging is similar to the difference between fully-associative and set-associative cache memories, where pages are allocated within a specific set (color) of possible pages.

The number of colors is an important parameter in this study. It is determined by the configuration of the memory system of the machine, specifically by the relative sizes of the cache memory and the physical memory pages. More details on this determination of number of colors, and the advantages of coloring, are available in [BLF90].

# 2 Methodology

A simple analytical model and trace-driven simulation both provide methods for studying performance differences between colored and uncolored (conventional) page allocation. These differences take the form of less effective memory utilization by thrashing on individual colors rather than a unified (uncolored) memory; if all of the pages were allocated into a few colors (i.e., the distribution of the pages over the colors is uneven), thrashing could occur on the heavily allocated colors while other colors are lightly used.

Two figures of merit are effective memory utilization and page-fault rate. Effective memory utilization measures the memory size at which a colored paging scheme would start to thrash on one color of physical memory (that is, when that color becomes fully allocated) relative to the size at which uncolored

paging thrashes (that is, the entire physical memory is fully allocated). This ratio is calculated by allocating pages until one color is fully allocated, and then dividing the total number of pages allocated by the total number of pages, and is necessarily less than or equal to one.

The other figure of merit, page-fault rate, is perhaps the more important one, as it is a direct factor in the overall execution speed of the machine. Page-fault rate is defined as the number of faults (the page is not in physical memory) divided by the number of memory references.

This study assumes that allocated pages are not later re-allocated into different physical locations, and that the initial ordering of pages on the free lists is random. These assumptions only approximate the behavior of real systems because reallocation does happen and the collection of free pages is somewhat orderly (e.g., proceeding around the physical memory in a clock-sweep algorithm [PS85]). However, the effects of these assumptions are considered small.

## 3 Analytical Model

The first method of analysis is an analytical model of the behavior of colored page allocation. The analytical model assumes that allocated virtual pages are randomly and uniformly distributed over all of the colored free lists.

A complicated analysis is required to determine the effective memory utilization. This analysis covers the case where the first colored free list fills up (the presumption being that thrashing is beginning on that free list), and determines the expected number of pages that have been allocated until that point.

This analysis is similar to the Birthday Surprise problem [KN67]. Calculating the expected number of pages allocated until any one colored free list is full gives

$$E(c, \mathrm{k}) = \int_{J0}^{\infty} [\mathrm{Sk}(t/c)]^{c} e^{-t} dt,$$

where

$$\mathrm{Sk}(Z) = \sum_{j=0}^{k} x^{j}/j!,$$

c is the number of colors (free lists), and $k$ is the number of available pages of each color (length of each free list). Thus, the effective memory utilization is given by the expected number of pages allocated until one free list is full divided by the total number of pages available (the number of pages which would be allocated before a single color system was full),

$$M_{\mathrm{eff}} = \frac{E(c, k)}{ck}.$$

Unfortunately, the approximations given [KN67] to $E(c, k)$ are valid only for $c \rightarrow \infty$, and are inappropriate for this paging interpretation of the problem.

However, numerical integration provides good approximate answers, which are validated with a few thousand simulator runs of the model (pages were allocated randomly over the colored free lists until one was full).

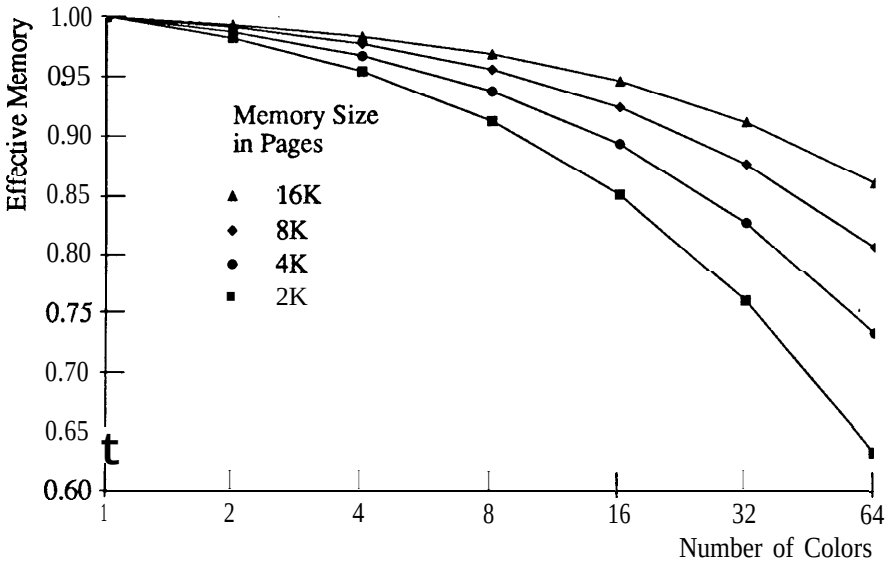


Figure 1: Modelled Effective Memory Utilization

Figure 1 shows the results of the numerical integration. As expected, the effective memory utilization falls as the number of colors increases. For a typical page size of 4 KB, these curves are for physical memory sizes of 8 MB through 64 MB. For either large memories or small numbers of colors, (organizations with many pages per free list) the modelled effective memory utilization is reasonable, staying above 90% for a variety of common configurations.

# 4 Trace-Driven Simulation

Trace-driven simulation checked the validity of the model's assumption of random requests for colors. An already existing simulator was modified to produce the desired information when fed address traces. In the simulator, in contrast to the model, pages can be allocated, freed through LRU replacement, and reallocated. The pages are, however, always allocated to the same color (i.e., the coloring function does not change).

To simulate paging systems such as this, address traces which touch many pages are required. Fortunately, some long traces from DECWRL [BKLW89]

| Name | Description | Unique Pages (4KB) |
|---|---|---|
| sor | Fortran implementation of sparse-matrix successive overrelaxation algorithm | 12416 |
| tree | Compiled Scheme building trees and searching for the largest element | 4518 |
| tv | Pascal timing verifier analyzing MultiTitan CPU | 5771 |
| lin | Circuit analysis program | 1158 |
| mult | multiprogramming workload (make, grr router, magic design rule checker, tree, xld, and continuous interactive shell commands) | 12480 |

**Table** 1: Simulation Traces

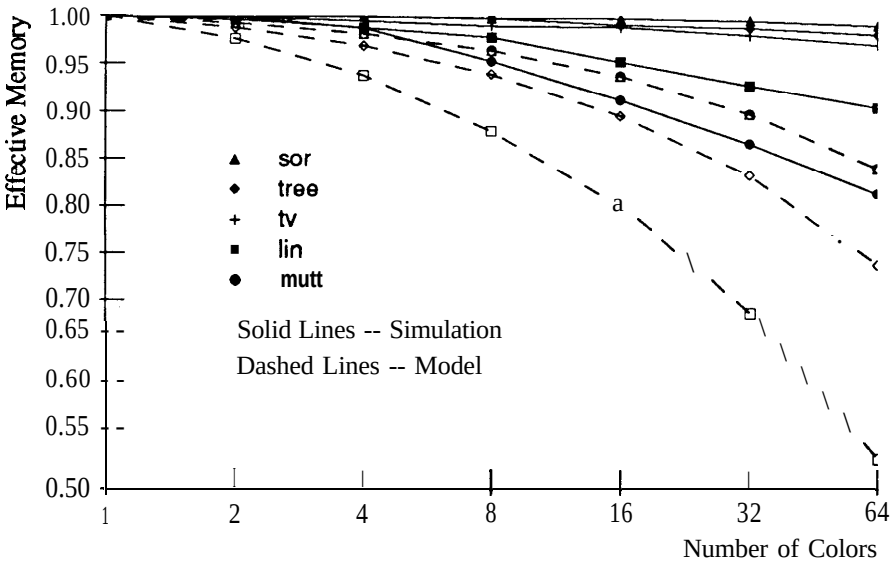


Figure 2: Simulated Effective Memory Utilization

were available to us. Technical limitations of the existing simulator limited simulations of the traces to between 1 and 1.5 billion references per trace. These traces were generated on a Titan, a 32-bit RISC machine running a version of Ultrix, and comprise several benchmarks [BKLW89]. Table 1 presents pertinent information about the traces. To effect a worst-case study, multrequests were not PID hashed, and thus a large number of processes all had stack, instruction, and data segments starting in pages of the same color.

Simulations provide results for both effective memory utilization, and page-fault rate. Page-fault rate results can also be gained through analytical models [Smi78], but their accuracy is unclear.

## 4.1 Effective Memory Utilization

Figure 2 presents the simulated effective-memory results. The analytical model was also evaluated for each benchmark using the number of pages in the program as the equivalent memory size, and the figure 2 also presents the model results for each benchmark size.

With the exception of **mult**, the benchmarks have less page-fault degradation due to coloring (i.e., are closer to 1 $M_{\text{eff}}$) the larger they are, and (again except for **mult**) stay above 90% $M_{\text{eff}}$ over the entire range shown. The exception is **mult**, which, as described earlier, actually is a large number of small programs, and thus has worse than uniform-random distribution of pages (i.e., some colors with many pages, some colors nearly empty).

The difference between simulated memory utilization of the benchmarks and that predicted by the analytical model indicates the color-distribution of pages not a uniform-random distribution, as assumed by the model. The assumption of random color requests is very pessimistic because typical allocation patterns tend to request pages somewhat sequentially in each of the text, data, and stack segments, resulting in request pattern closer to round-robin than random for each segment, and a controlled and even distribution of pages per color.

## 4.2 Page-Fault Rate

While effective memory utilization is a reasonable measure for relating colored page allocation to uncolored page allocation,' page fault rate is the significant measure. Page fault rate, while difficult to model accurately, can be easily simulated, so simulations were run to garner page-fault rates. Figures 3-7 present these page-fault rate results. (Note that in these page-fault rate graphs, a lower page-fault rate indicates better performance, as opposed to the effective memory utilization graphs, where a higher effective-memory utilization was better.)

Figures 3-7 give the page-fault rates for main memory sizes from 4 MB to 64 MB, and from 1 to 64 colors. These page-fault rates are plotted relative to the uncolored (l-color) page-fault rate for each memory size (hence all l-color memory sizes show relative fault rate 1). Tables 3-6 in Appendix A present the

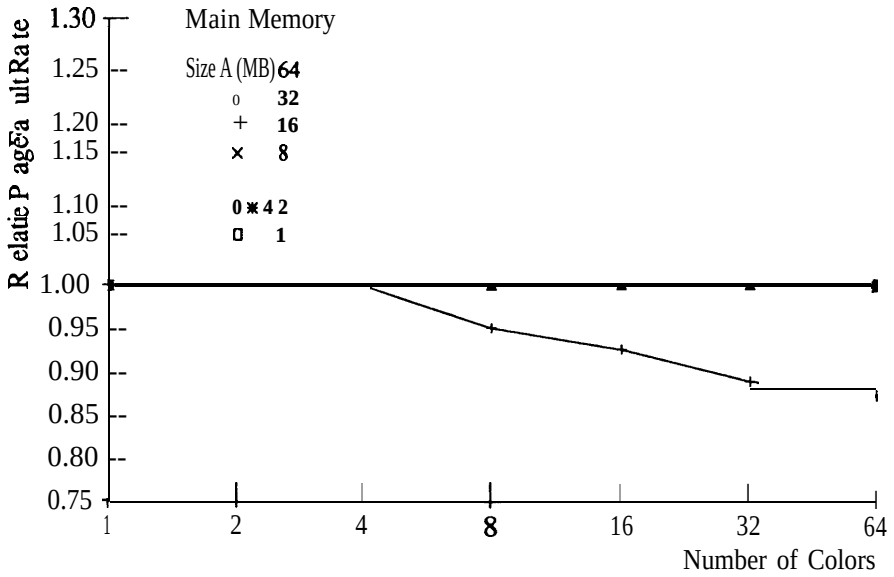


Figure 3: Page-Fault Rate – sor

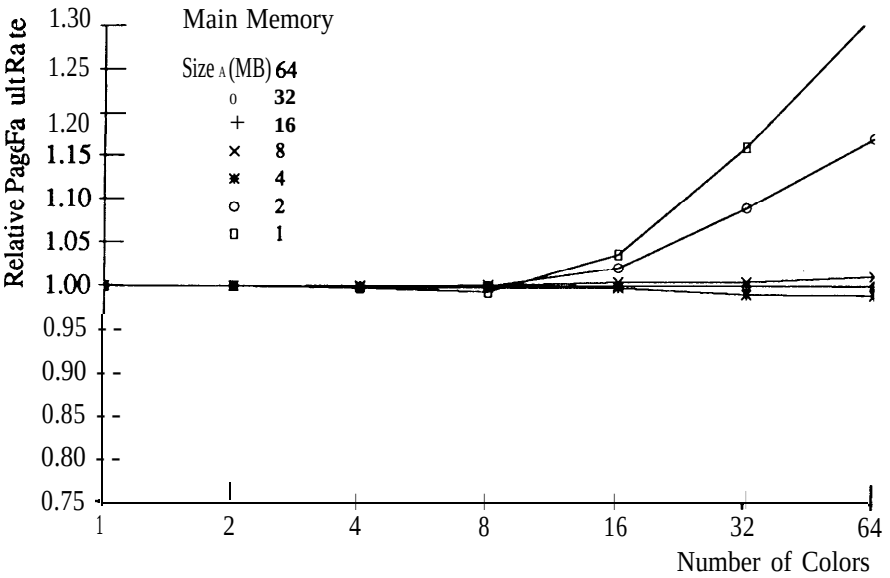


Figure 4: Page-Fault Rate – tree

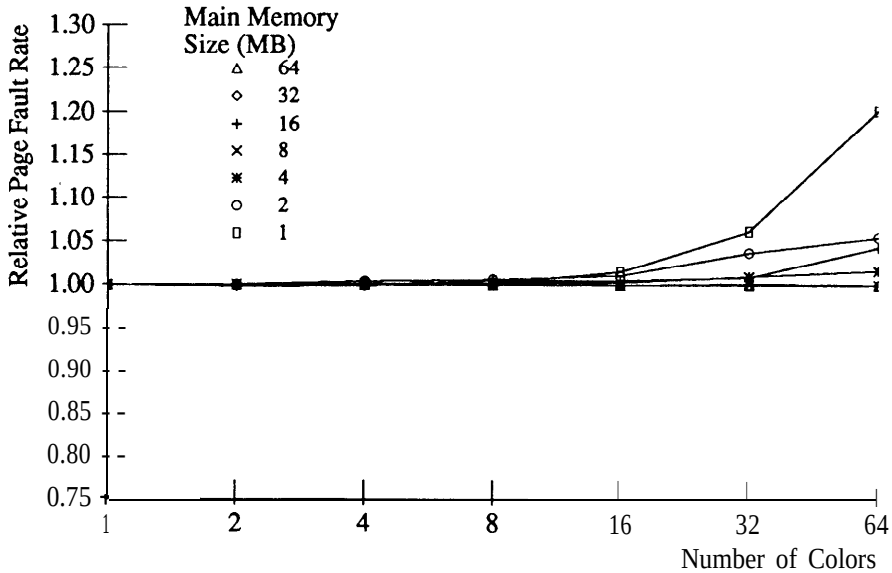


Figure 5: Page-Fault Rate – tv

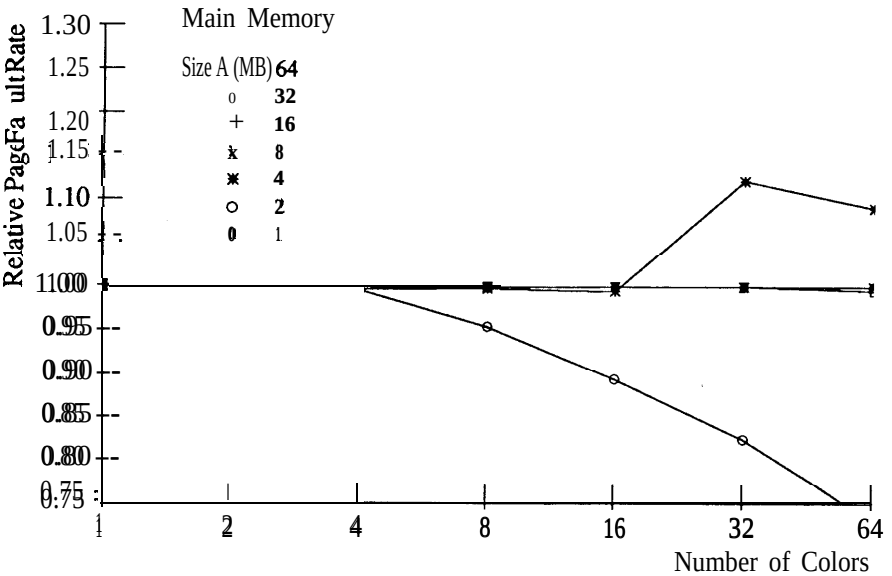


Figure 6: Page-Fault Rate – lin

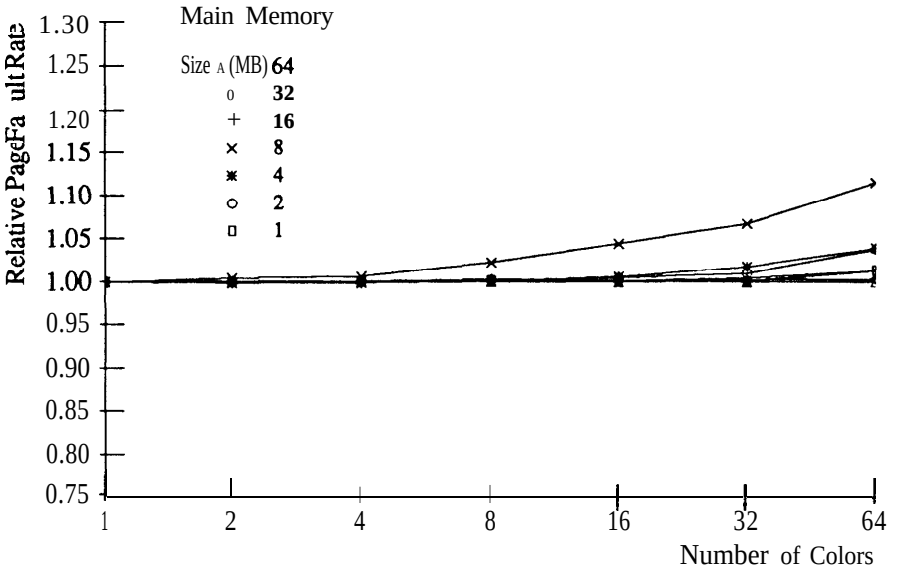


Figure 7: Page-Fault Rate - `mult`

same results in a non-relative tabular form. The 1 MB, 2 MB, and 4 B memory systems are included for completeness, but note that machines of this size are too small to effectively run benchmarks of the size used here.

The results clearly show that page-fault rates are not affected much by the number of colors. At most, the effect is pronounced only for one memory size per benchmark (e.g., 16 MB sor). Surprisingly, the results with significant effect were often advantageous (i.e., a lower page-fault rate), as in sor16 MB, and `lin2` MB.

It might be expected that the page-fault rate would rise as the number of colors rose and the memory became fragmented into smaller and smaller sections. This unexpected reduction in page-fault rate results from the frequently sequential access nature of programs of this size. Most of the pages accessed are on the heap and belong to large data structures, which programs typically access by sweeping through them. This sweeping access pattern causes the pathological behavior of fully-associative LRU replacement policies, where $n + 1$ elements accessed in rotation in $n$ slots always fault, while set-associativity causes some elements to remain and not fault.

Even these unlikely cases where the page-fault rate decreases with increasing numbers of colors are rare results in these simulations. If the 1 MB, 2 MB, and 4 MB physical memory sizes are disregarded as unreasonable for benchmarks of

this size, only 4 of 20 combinations of benchmarks and physical-memory sizes changed by more than 1% from the l-color (conventional) to 64-color case. Of those 4, 2 had increased fault rates (tv, 5%, and `mult`, 12% increases at 64-colors) and 2 had decreased fault rates (`lin`, 30%, and sor, 13% decreases at 64 colors).

Other than very few cases with significant deviation in page-fault rates, the change in page-fault rate with the number of colors was less than 1% for the region simulated.

# 5 Conclusions

This paper presents an analytical model demonstrating the effective memory utilization of colored paging. The model, while pessimistic, does provide the right form for the effective memory utilization curves, and "random simulations" of the model validate its accuracy under the random allocation assumption. However, this assumption of uniformly-distributed color requests is very pessimistic compared to typical allocation patterns, which generally tend to request pages sequentially resulting a more even distribution of pages per color than a random request pattern.

This view of program behavior is supported by simulations results, where the page-fault rate does not noticeably increase as the number of colors increased. Just as often, the benchmarks studied show improved (decreased) page-fault rates with increased coloring. Far more likely than either direction of change in page-fault rates, however, is for the page-fault rate not to change significantly. Therefore, the basic conclusion of this study is that over a broad range of machine configurations the use of colored paging has no deleterious performance effects through extra paging activity.

# A Raw Simulation Results

This appendix presents tabular results of page-fault rate for simulations of each of the benchmarks sor, tree, tv, `lin`, and `mult`. The page size for these simulations is 4KB. Large numbers of colors perform significantly worse for the small memory sizes as the number of pages per free lists becomes small (for example, 1 MB memory, 64 colors gives a mere 4 pages of each color). Memory sizes of 1 MB, 2 MB, and 4 MB are presented here only for completeness, as they are not appropriately-sized machines for these problems.

| Memory Size (MB) | Number of Colors | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 1 | 9146.10 | 9145.37 | 9147.95 | 9147.91 | 9148.50 | 9150.10 | 9163.79 |
| 2 | 7388.46 | 7388.52 | 7390.94 | 7390.13 | 7390.69 | 7392.90 | 7391.68 |
| 4 | 3872.64 | 3873.15 | 3874.53 | 3873.60 | 3874.10 | 3874.68 | 3873.22 |
| 8 | 45.27 | 45.28 | 45.28 | 45.28 | 45.26 | 45.26 | 45.21 |
| 16 | 41.14 | 41.14 | 41.14 | 39.10 | 38.07 | 36.54 | 35.93 |
| 32 | 31.27 | 31.28 | 31.28 | 31.27 | 31.27 | 31.26 | 31.27 |
| 64 | 8.11 | 8.11 | 8.11 | 8.11 | 8.11 | 8.11 | 8.11 |

Table 2: Page-Fault Rate (x $10^{-6}$) – sor

| Memory Size (MB) | Number of Colors | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 1 | 122.79 | 122.80 | 122.53 | 121.97 | 127.19 | 142.35 | 161.25 |
| 2 | 13.69 | 13.69 | 13.68 | 13.69 | 13.97 | 14.92 | 16.00 |
| 4 | 11.43 | 11.43 | 11.42 | 11.41 | 11.40 | 11.32 | 11.31 |
| 8 | 6.49 | 6.49 | 6.49 | 6.50 | 6.52 | 6.52 | 6.56 |
| 16 | 4.39 | 4.39 | 4.39 | 4.39 | 4.39 | 4.39 | 4.39 |
| 32 | 3.90 | 3.90 | 3.90 | 3.90 | 3.90 | 3.90 | 3.90 |
| 64 | 3.90 | 3.90 | 3.90 | 3.90 | 3.90 | 3.90 | 3.90 |

Table 3: Page-Fault Rate (x $10^{-6}$) – tree

| Memory Size (MB) | Number of Colors | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 1 | 124.15 | 124.11 | 124.34 | 124.36 | 126.00 | 131.75 | 149.14 |
| 2 | 89.20 | 89.28 | 89.59 | 89.75 | 90.14 | 92.49 | 94.08 |
| 4 | 69.14 | 69.19 | 69.23 | 69.30 | 69.35 | 69.80 | 70.27 |
| 8 | 50.64 | 50.66 | 50.68 | 50.67 | 50.62 | 50.71 | 50.64 |
| 16 | 17.86 | 17.83 | 17.84 | 17.95 | 17.95 | 18.01 | 18.63 |
| 32 | 3.82 | 3.82 | 3.82 | 3.82 | 3.82 | 3.82 | 3.82 |
| 64 | 3.82 | 3.82 | 3.82 | 3.82 | 3.82 | 3.82 | 3.82 |

Table 4: Page-Fault Rate (x $10^{-6}$) – tv

| Memory Size (MB) | Number of Colors | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 1 | 94.27 | 94.26 | 94.27 | 94.25 | 94.27 | 94.27 | 93.83 |
| 2 | 93.83 | 93.83 | 93.44 | 89.44 | 83.74 | 77.21 | 68.45 |
| 4 | 1.01 | 1.01 | 1.01 | 1.01 | 1.00 | 1.13 | 1.10 |
| 8 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 |
| 16 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 |
| 32 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 |
| 64 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 |

Table 5: Page-Fault Rate (x $10^{-6}$) – `lin`

| Memory Size (MB) | Number of Colors | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 1 | 750.47 | 750.69 | 750.99 | 751.41 | 750.67 | 750.80 | 760.01 |
| 2 | 489.83 | 490.25 | 490.11 | 490.84 | 492.06 | 494.53 | 507.97 |
| 4 | 232.90 | 232.56 | 232.55 | 233.07 | 234.27 | 236.88 | 241.97 |
| 8 | 55.18 | 55.41 | 55.50 | 56.40 | 57.61 | 58.89 | 61.50 |
| 16 | 14.13 | 14.13 | 14.13 | 14.17 | 14.14 | 14.19 | 14.31 |
| 32 | 10.09 | 10.09 | 10.09 | 10.10 | 10.10 | 10.11 | 10.12 |
| 64 | 9.73 | 9.73 | 9.73 | 9.73 | 9.73 | 9.73 | 9.73 |

Table 6: Page-Fault Rate (x $10^{-6}$) – `mult`

## References


[BKLW89] Anita Borg, R. E. Kessler, Georgia Lazana, and David W. Wall. Long Address Traces from RISC Machines: Generation and Analysis. WRL Research Report 89/14, Digital Equipment Corporation, September 1989.

[BLF90] Brian K. Bray, William L. Lynch, and M. J. Flynn. Page Allocation to Reduce Access Time of Physical Caches. Technical Report CSL-TR-90-454, Stanford University, November 1990.

[KN67] M. S. Klamkin and D. J. Newman. Extensions of the Birthday Surprise. ***Journal of Combinatorial Theory,*** 3:279–282, 1967.

[PS85] James L. Peterson and Abraham Silberschatz. *Operating* ***System*** Concepts. Computer Science. Addison-Wesley, Reading, MA, 1985.

[Smi78] Alan J. Smith. A Comparative Study of Set Associative Memory Mapping Algorithms and Their Use for Cache and Main Memory. ***IEEE Transactions on Software Engineering,*** SE-4(2):121-130, March 1978.