March 2003: This file was created by scanning, OCR, and touchup of one of the originally-distributed paper copies.

M0131

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

November 3, 1972

A SIMPLE LINEAR MODEL OF DEMAND PAGING PERFORMANCE

by

Jerome H. Saltzer
Room 519
545 Technology Square
Cambridge, Mass. 02139

Abstract: Predicting the performance of a proposed automatically managed multilevel memory system requires a model of the patterns by which programs refer to the information stored in the memory. Some recent experimental measurements on the Multics virtual memory suggest that, for rough approximations, a remarkably simple program reference model will suffice. The simple model combines the effect of the information reference pattern with the effect of the automatic management algorithm to produce a single, composite statement: the mean number of memory references between paging exceptions increases linearly with the size of the paging memory. The resulting model is easy to manipulate, and is applicable to such diverse problems as choosing an optimum size for a paging memory, arranging for reproducible memory usage charges, and estimating the amount of core memory sharing.

Key Words and Phrases: paging, demand paging, memory models, program models, performance measurement, multilevel memory systems, virtual memory, associative memory, memory usage accounting, Multics

Computing Reviews categories: 3.79,4.30, 4.32

Work reported herein was conducted by the Computer Systems Research Division of Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract number N00014-70-A-0362-0006.

Introduction

Research in engineering includes identifying successively more precise, but often less tractable, models of real-world phenomena. Although it is only occasionally that a mathematically tractable model happens to exactly represent the real-world situation, often an approximate model is good enough for many engineering calculations. The challenge in constructing approximate models is to maintain mathematical tractability in the face of obvious flaws and limitations in the range of applicability and yet produce a useful result. This paper proposes, reports a level of confidence in, and analyzes what is probably the simplest model of a demand paged virtual memory computer system.

The Model

Prediction of performance of a proposed configuration of an automatically managed multilevel memory requires a model of the patterns by which programs refer to the information stored in the memory. If one has sample reference strings, it is straightforward to simulate the performance of a proposed memory configuration using the sample strings.[1] However, one would prefer to work with a more compact representation of the reference pattern than an exhaustive trace. Some recent experimental measurements on the Multics virtual memory suggest that, for rough approximations, a remarkably simple program reference model will suffice. The simple model combines the effect of the information reference pattern with the effect of

the automatic memory management algorithm to produce a single, composite statement: the mean number of memory references between paging exceptions increases linearly with the size of the paging memory.

We may make this statement of the simple model more precise with the aid of figure one. The central processor is seen as the source of a string of references to a virtual memory which is implemented by an automatically managed (e.g., demand paged) two-level memory system. The automatic management may be by hardware, as in the buffer memory of the IBM System 370/195, in which case memory level one is bipolar silicon transistor memory, and memory level two is core memory. Alternatively, the automatic management may be by software, as in the Multics virtual memory, in which case memory level one is currently core memory, and memory level two currently consists of magnetic drum and disk memories. As seen below, a memory of three or more automatically managed levels may be modelable as a two-level memory by drawing appropriate boundaries. The simple model states that the mean number of consecutive references to memory level one before something is demanded from memory level two (we call this quantity the mean headway between page exceptions) is linearly proportional to the size of memory level one. If $\bar{r}$ is the mean headway between page exceptions and Z_1 is the size of memory level one, we have

$$\bar{r} = aZ_1 \qquad (1)$$

providing a simple model with a single parameter, a.

A Confidence Test

Two empirical measurements are offered in support of the contention that this crude model is sufficiently realistic to permit at least rough calculations. First, the Honeywell 645 central processor contains a small, hard-

ware-managed associative ("look-aside") memory which retains frequently used page table words using a Least-Recently-Used (LRU) algorithm to choose which entries should be displaced by new ones. We may regard this associative memory as memory level one, and the actual page maps, in core memory, as memory level two. Figure two indicates the observed mean headway between associative memory "no-match" responses as a function of associative memory size, as reported by Schroeder.[2] It is apparent that, within the range of Schroeder's measurements, a linear model provides an approximation to the observed behavior, represented by circles.

A second, previously unpublished, set of measurements characterizes the demand paging operation of Multics.[3,4] Multics at M.I.T. is operated at different times with two core memory sizes. The paging area is managed with an algorithm approximating LRU*. The system continuously records summaries of paging activity, and of total time accumulated in each of three major categories: "paging", "idle", and "useful". "Paging" time is accumulated whenever a page exception is being processed by software. "Idle" time accumulates whenever there is nothing a processor can do (either the processor queues are empty, or all jobs being multiprogrammed are waiting for something to be moved from memory level two.) Finally, "useful" time is that time constructively utilized by users of the virtual memory. By counting the total

---

* When a page is brought into memory level one, it is placed at the end of a linked list with its hardware usage bit turned off after the initial reference which brought it into memory. When a page is to be selected for removal from memory the page at the front of the linked list is examined. If its usage bit is off, it is selected for removal. If its usage bit is on, it is turned off, that page is moved to the end of the list, and the new front page is similarly examined. (This description is slightly simplified. Reference [4] should be consulted for full detail.)

number of paging exceptions in, say, a two-hour period, and dividing that number into the amount of useful time accumulated in the same period, we obtain a suitable mean headway between page exceptions. This figure, measured in units of memory references to allow comparison with other systems, is plotted in figure three for two typical several-hour operating periods with different memory sizes. The load on the system at those times was an uncontrolled daytime population of thirty to fifty M.I.T. users. Although the lack of control may seem worrisome, repeated similar experiments almost always produce results within about 10% of the typical results of figure three.

If the straight line of figure three were extrapolated to the right as far as 4,000 pages, it would predict a headway between page exceptions of about 90,000 references. In fact, Multics uses an automatic three-level memory system, with a drum of 4,000 pages at the second level, disk memories at the third level, and an LRU algorithm governing placement between the disk and the drum. A typical observed headway between disk references with that drum size is 85,000 virtual memory references, which is in the general vicinity predicted by the linear model. Further, when the drum size is halved to 2,000 pages, disk traffic approximately doubles.

To explore the effect of drum size more carefully, a specially instrumented drum control package was developed. An LRU stack for the Multics drum is implemented as a chained pointer list in memory level one. Whenever a drum page is referenced, its entry on the LRU stack is moved to the top of the stack by restringing it at the front of the chained pointer list; when a page must be pushed off the drum the bottom one in the stack is chosen. The added instrumentation consists of noting, whenever a page

is moved to the top of the LRU stack, its stack depth, and accumulating the number of references at each stack depth. Because LRU is a stack algorithm[l] one can directly calculate how many additional references to the disk memory level would have occurred for each possible smaller drum size, by adding up the number of references to stack depths greater than the suggested smaller drum size.* Thus a single set of stack depth reference counts allows one

---

* This technique does not permit extrapolation to drum sizes smaller than the size of memory level one since the drum maintains copies of pages in level one, the top of the drum stack contains only those copies, and most references to those pages (which are satisfied in level one) will not cause the drum stack to be updated.

---

to calculate the mean headway between disk references which would have occurred for every possible drum size. Figure four is a typical result of this measurement and calculation, showing mean headways for drum sizes between 320 and 2,048 pages, at intervals of 64 pages. The statistics of figure four represent approximately 63 minutes of running time, with 260,000 page movements from drum and disk to core. The system load, as usual, was thirty to fifty uncontrolled time-sharing users. The measurements of figure three are superimposed near the lower left corner of figure four, for comparison. From figures three and four -- and the experience with a 4,000 page drum--it is evident that a linear model provides an approximate description of the paging behavior of Multics over a 40:1 range of memory sizes. Measurements based on detailed tracing of all disk memory references are underway, to explore the shape of the mean headway function in the region between 4,000 pages and 75,000 pages, the current size of the on-line information base at M.I.T.

At least two sets of published measurements[5,6] report mean headway between page exceptions increasing much faster than linearly with memory size. One difference between those measurements and the ones reported here is evident: the programs in those measurements had been designed to operate in the fixed (and relatively small) memory of the IBM 7094 or AN/FSQ7 computers; the experiments consisted of "squeezing" these programs into spaces smaller than the ones for which they had been designed. In contrast, essentially all Multics programs have been written with the paged virtual memory environment in mind. The authors of these programs have not attempted to fit their programs into some arbitrary size of memory; instead they used as much virtual memory as they felt was economical for the task at hand. Although other related measurements have also been published[7,8,9] not enough information was given to easily relate those measurements to the ones reported here.

A second, perhaps more important, difference between the measurements reported here and the others is that the Multics measurements are not of a single program but rather of a system which is dynamically multiprogramming a variable number of individual programs. Thus a variety of averaging and smoothing effects are undoubtedly in operation. The relation between the linear model, as applied to a system and, e.g., Denning's working set model[10], as applied to a single program, is not immediately apparent.

Applications of the model

The simple linear model is especially interesting because of the ease of mathematical analysis when applying it to a variety of rough engineering calculations. Belady and Kuehner[11] made the same observation in 1969, but measurements available at that time led them to study more complex,

exponential models. The following four examples illustrate just how far one can go using the simpler, linear model:

1. Calculating the most cost-effective size for a memory level.
2. A memory usage charge which is independent of real memory size.
3. Analysis of a three (or more) level memory.
4. Estimation of the amount of sharing, or working set overlap.

Consider first the estimation of the most cost-effective size for memory level one in a two level automatically managed memory. If we label the average access time to the first level τ_1 and to the second level τ_2 we will have an overall average access time, τ_m of approximately*

$$\tau_m = \tau_1 + \tau_2/az_1 \qquad (2)$$

since, by the linear model, the relative frequency of references to memory level two is $1/az_1$. If we assume that the overall system performance is linearly proportional to the average memory access time, and that the system rental cost is partitioned into two parts, C_p, which is independent of the size of memory level one, and z_1C_m, which is proportional to the size of memory level one, we have a cost per reference of

$$\text{Cost} = (C_p + z_1C_m)\ (\tau_1 + \tau_2/az_1) \qquad (3)$$

This function has a minimum with respect to Z1 at

$$z1 = \sqrt{(\ \tau_2/\tau_1 * C_p/C_m * 1/a\ )} \qquad (4)$$

allowing an estimate of the most cost-effective size for the first level memory.

---

* We assume, for simplicity, that an access to memory level one is needed for every reference. We also assume that any processing cost of a page exception is included in τ_2.

This formula predicts, for example, that for the Honeywell 645 computer system, for which τ_2 =1 15 x 10^{-3} sec., τ_1 = 1.5 x 10^{-6} sec., c_p = \$16,000/mo., c_m = \$160/mo/page, and a = 20* (from figure 4), the optimum primary memory size with a single processor should be about 225 pages. This prediction correlates well with the empirical observations that with 200 pages and one processor, Multics appears to be memory-limited, while with 300 pages the one processor system appears to be processor-limited. A similar application of this formula can be used to estimate the optimum size of the associative memory for page table words used in most virtual memory processors.

A second application for the linear paging model lies in developing a charge for primary memory usage which is independent of the size of the physical memory which happens to be in use at the time. The method is to extrapolate the observed linear behavior of the entire system to the behavior of a single program. (One must expect that such an extrapolation is at best a rough approximation.) If a job requiring R virtual memory references is run with a paging area of size Z_1, the linear model predicts that the number of page exceptions observed will be R/a_jZ_1. The value of a_j is subscripted to indicate that it is a measure of the mean headway between page exceptions caused by this job alone. The combination R/a_j is a useful measure of the resource load of the job -- it increases with the length of the job, R, and it decreases with decreasing paging activity by the job, as measured by a_j. To obtain an estimate of R/a_j for a running job, we can simply count the number of page exceptions (which should be approximately R/a_jZ_1) and multiply that number by Z_1, the size of the paging area in use at the time.

---

* The units of a are inverse pages.

If a charge is made proportional to this observed value of R/a_j, it will provide an incentive to the user to increase the locality of reference of the job, thereby increasing the value of a_j and the mean headway between page exceptions, and thus reducing the paging load on the system. On the other hand, if the job is operated with a smaller or larger paging area, because of configuration or load changes, the charge to the user will remain approximately constant to the extent that the linear model applies. Such configuration- and load- independent measures have been requested by many users[12] to simplify accounting and budgeting; they also permit the installation to optimize overall system effectiveness by adjusting the level of multiprogramming and therefore the average size of the paging areas, without fear of job costs changing.

A memory charging scheme based on these observations has been experimentally added to the Multics system at M.I.T. Since Multics is multiprogrammed, this scheme involves estimating the size of the paging area available to the user by dividing the total paging area by the number of processes being multiprogrammed. Each time a page exception occurs, the paging area size is estimated and added to a running total for the process. When the process finishes, this running total is proportional to the value of R/a_j, and an appropriate charge is computed. Initial experiments with this strategy indicate that when the size of the paging area available to a single program is varied by more than a factor of ten, the resource usage, as measured by the estimate of R/a_j, varies no more than 30 to 50%. This variation is a measure of the extent to which an individual program does not exactly follow a linear model.

A third application for the linear model is the analysis of an automatically managed memory system of three or more levels as in figure five. If we assume that each level contains a copy of information in the next higher (and smaller) level, then the linear model predicts that the mean number of virtual memory references before a reference to level j will be (aZ_{j-1}), since Z_{j-1} represents the effective size of the memory down to and including level (j-1). Similarly, the mean number of virtual memory references before a reference to level (j+1) will be (aZ_j). The fraction of incoming memory references which fail to be satisfied by level j is thus $(Z_{j-1})/Z_j)$. This formula permits estimation of necessary channel capacities between the memory devices as well as overall memory system average access time. Even more important, if one develops a model of access times and costs such as that of equation (3) for a three level memory, one may calculate, for example, whether or not one should discard the middle level and invest instead in a larger memory level one. Such a calculation is applicable, for example, to the question of whether future systems should be based on a two level, LSI/videotape technology, or a third, (presumably disk storage) level will be needed.

Finally, we may use the linear model to estimate the amount of information sharing which occurs in a multiprogram or multiprocessor environment. The problem here is that, in systems such as Multics, which permit any page to be simultaneously used by any of several different processors, the benefit in reduced memory space required may be difficult to measure.

Consider two processors which share a single two-level memory. A single processor using the entire memory for itself has an observed mean headway between page exceptions of r_1 virtual memory references, as in figure six. Suppose that there is one job in memory for each processor, with memory more or less evenly divided among the jobs. Then with two processors, and if no sharing of pages in memory level one occurs, each of the central processors might be expected to exhibit about the same mean headway between page exceptions as does a single processor operating with half the total memory, namely $r_1/2$ virtual memory references. If sharing of pages does occur, the effect will be to increase the actual observed mean headway between page exceptions to some larger value, r_2. Again using the linear model, we may directly calculate the size of the effective overlap of the working sets of the two processors to be $(r_2-r_1/2)/a$ pages. The size of the overlap may be interpreted as the amount of level one memory saved by use of opportunities to share pages. A similar analysis can be made for a single processor performing multiprogramming on jobs with overlapping working sets. Measurements of Multics using this approach are planned, but have not yet been carried out.

Limitations of the Linear Model

The linear paging model must be viewed in the proper perspective -- it is not a panacea for memory system engineering problems. It has been only roughly validated by a few points measured on a single demand paging system. Clearly, the model cannot hold for extreme values, since at the size for which everything fits in primary memory, the mean headway between references to the secondary memory should become singular; the linear model does not predict this effect. Analysis is

needed to discover what microscopic properties of program reference strings and page selection algorithms might lead to the macroscopic behavior observed in the measurements; such analysis should provide a direct calculation of the value of the parameter, a.

It may be that the experimentally observed linearity is a special property of the LRU algorithm, which was in use in every experiment reported here, or of some interaction of the LRU algorithm and the particular reference patterns of users of the Multics system. Perhaps more measurements on a wider variety of systems will reveal whether or not the model has any wider application, or can be used for more than "back of the envelope" initial guesses about paging system performance. In any case, when one is trying to predict within a factor of two the appropriate size for a memory level (and few systems today are engineered with much greater precision) the estimate of a linear approximation supported by a rough measurement of the value of "" for the load in question may be far more useful than having no model at all.

Acknowledgements

Akira Sekino developed a hierarchical strategy for system modeling which exposed the usefulness of a model of the mean headway between missing pages.[13] Professor F. J. Corbato suggested that a linear model would be remarkably easy to manipulate, and almost inevitably a satisfactory approximation at some level. Robert M. Frankston helped develop the use of the linear model as a device for estimating load independent core memory use charges. Akira Sekino helped investigate the linear model as a basis for estimating sharing among parallel processes using the same virtual memory. Michael D. Schroeder provided hardware measurements of the Honeywell 645 associative memory, and Steven H. Webber invented and installed meters for measurement of the Multics virtual memory.

References

[1] Mattson, R.L., et a1., "Evaluation Techniques fur Storage Hierarchies," IBM Systems Journal 9, 2 (1970), pp. 78-117.

[2] Schroeder, M.D., "Performance of the GE-645 Associative Memory While Multics is in Operation," ACM Workshop on System Performance Evaluation, Harvard University, (April 5-7, 1971), pp. 227-245.

[3] Bensoussan, A., Clingen, C.T., and Daley, R.C., "The Multics Virtual Memory: Concepts and Design," Communications of the ACM 15, 5 (May, 1972), pp. 308-318.

[4] Corbató, F.J., "A Paging Experiment with the Multics System," in Ingard, In Honor of P.M. Morse, M.I.T. Press, Cambridge, Mass., 1969, pp. 217-228.

[5] Varian, L.C., and Coffman, E.G., "An Empirical Study of the Behavior of Programs in a Paging Environment," ACM Symposium on Operating System Principles, Gatlinburg, Tennessee, (October 1-4, 1967).

[6] Fine, G.H., Jackson, C.W., and McIsaac, P.V., "Dynamic Program Behavior under Paging," ACM Proceedings of 21st National Conference, P-66, Thompson Books, Washington, D.C., 1966, pp. 223-228.

[7] 0'Neill, R.W., "Experience using a Time-shared Multi-programming system with Dynamic Address Relocation Hardware," AFIPS Conf. Proc. 30, (1966 SJCC), Thompson Books, Washington, D.C., pp. 611-621.

[8] Belady, L. A., "A study of Replacement Algorithms for a Virtual-storage computer," IBM Systems Journal 5, 2 (1966), pp. 78-101.

[9] Baylis, M.H.K., et a1., "Paging Studies Made on the I.C.T. Atlas Computer," Proc. IFIP Congress 68, North Holland Publ. Co., Vol. 2, pp. 831-837.

[10] Denning, P.J., "The Working Set Model for Program Behavior," Comm. ACM 11, 5 (May, 1968) pp. 232-333.

[11] Belady, L.A., and Kuehner, C.J., "Dynamic Space-Sharing in Computer Systems," Comm. ACM 12, 5 (May 1969), pp. 282-288.

[12] Taylor, A., "Should the Charges Vary with Each Job Execution?" ComputerWorld 5, 44 (November 3, 1971), p. 7.

[13] Sekino, Akira, "Performance Evaluation of Multiprogrammed Time-Shared Computer Systems," Ph.D. Thesis, M.I.T., Department of Electrical Engineering, in progress.

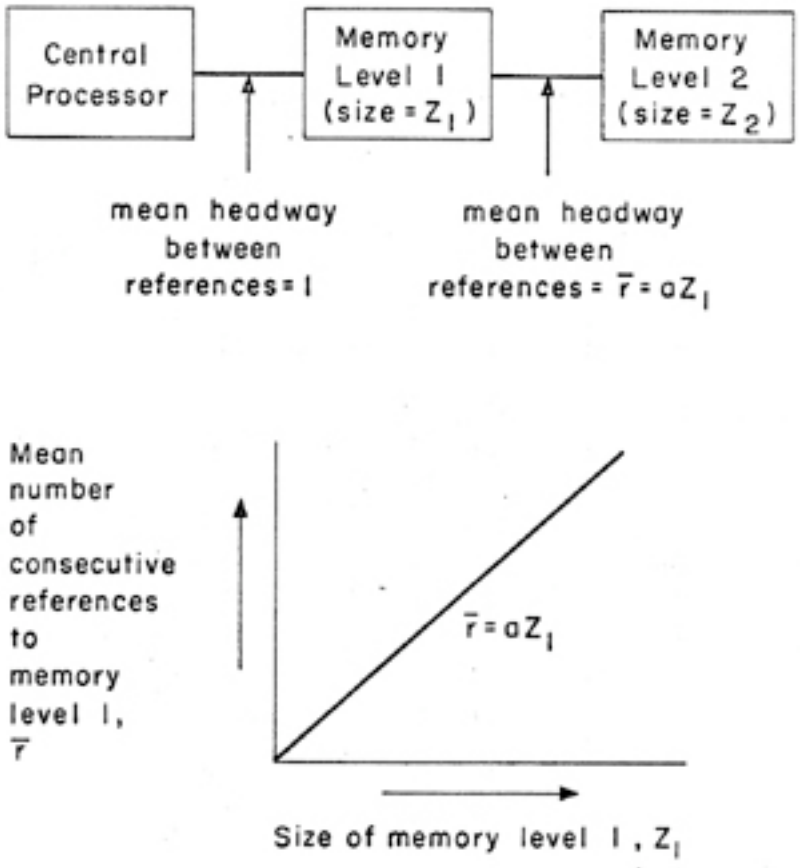

Figure 1: The simple linear paging model.

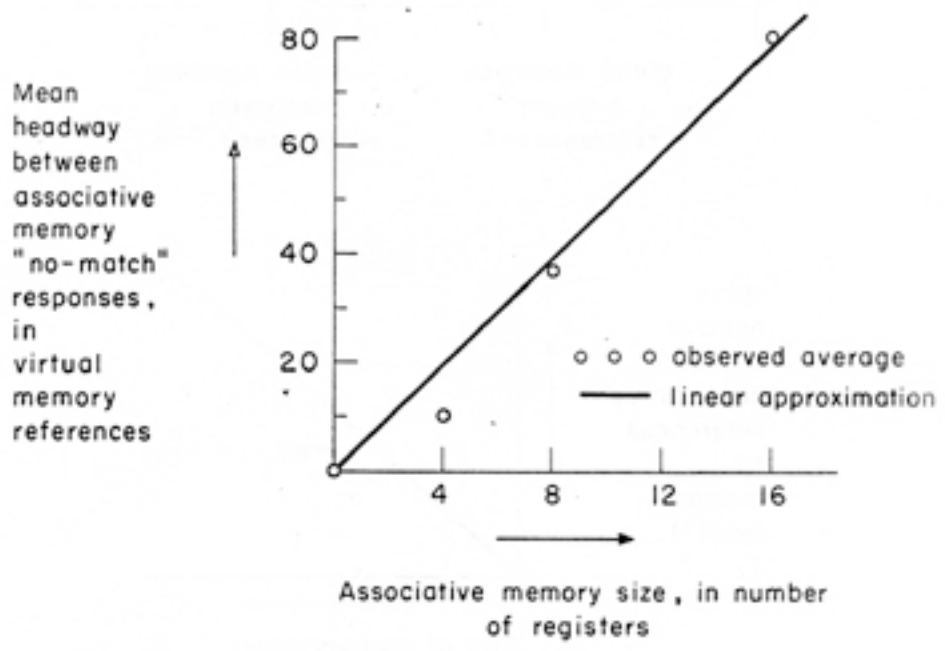

Figure 2: Associative memory performance of the Honeywell 645 computer.

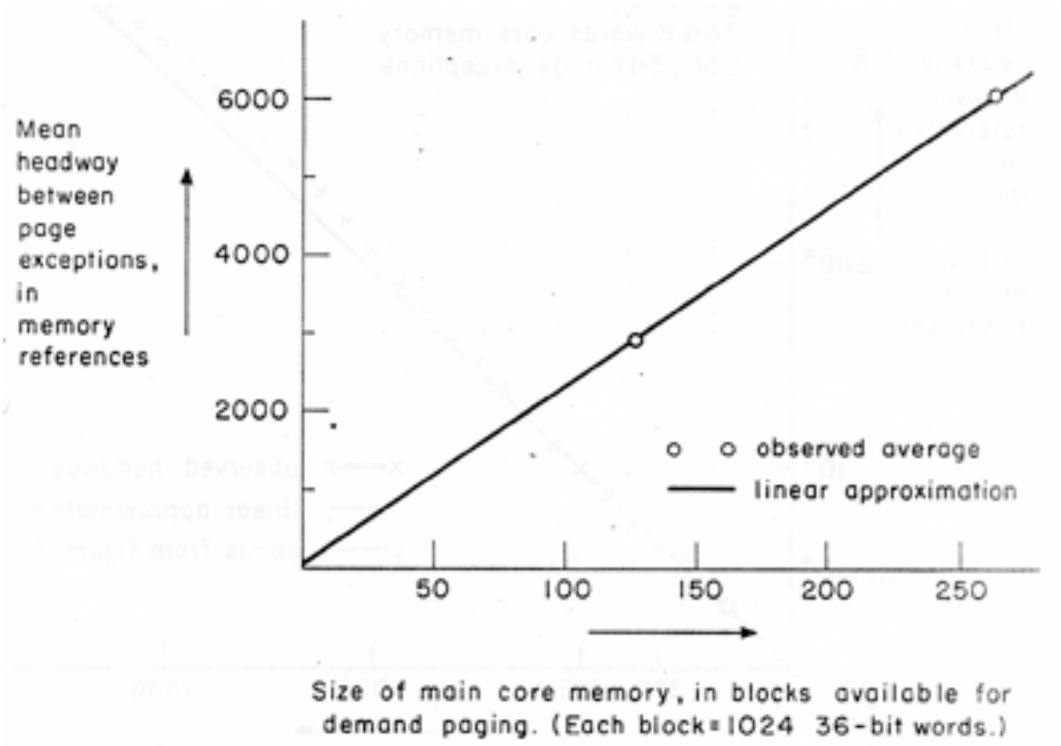

Figure 3: Mean headway between page exceptions in Multics for two memory sizes.

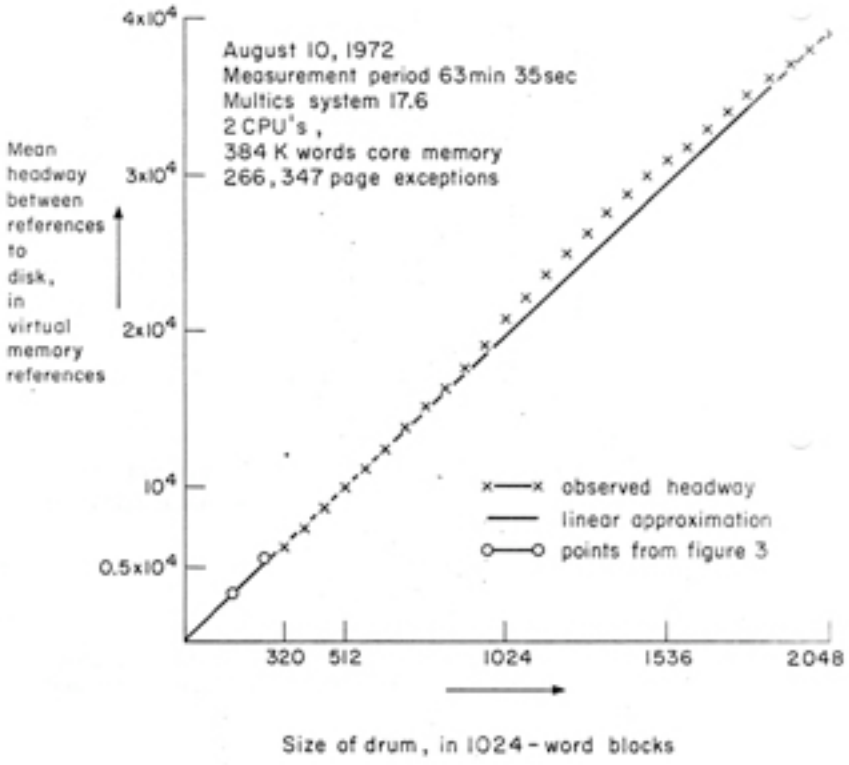

Figure 4: Mean headway between disk references in Multics, for 32 drum sizes, based on LRU stack depth measurements. The two points at the lower left are from figure three.

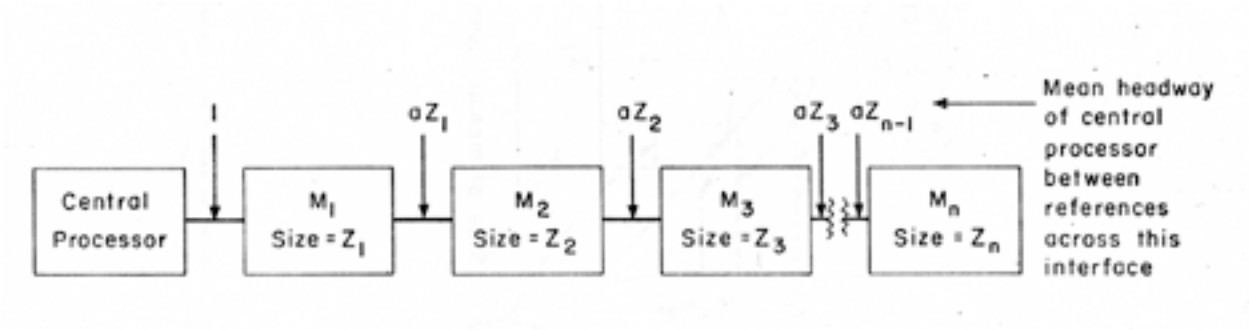

Figure 5: A general multilevel memory system.

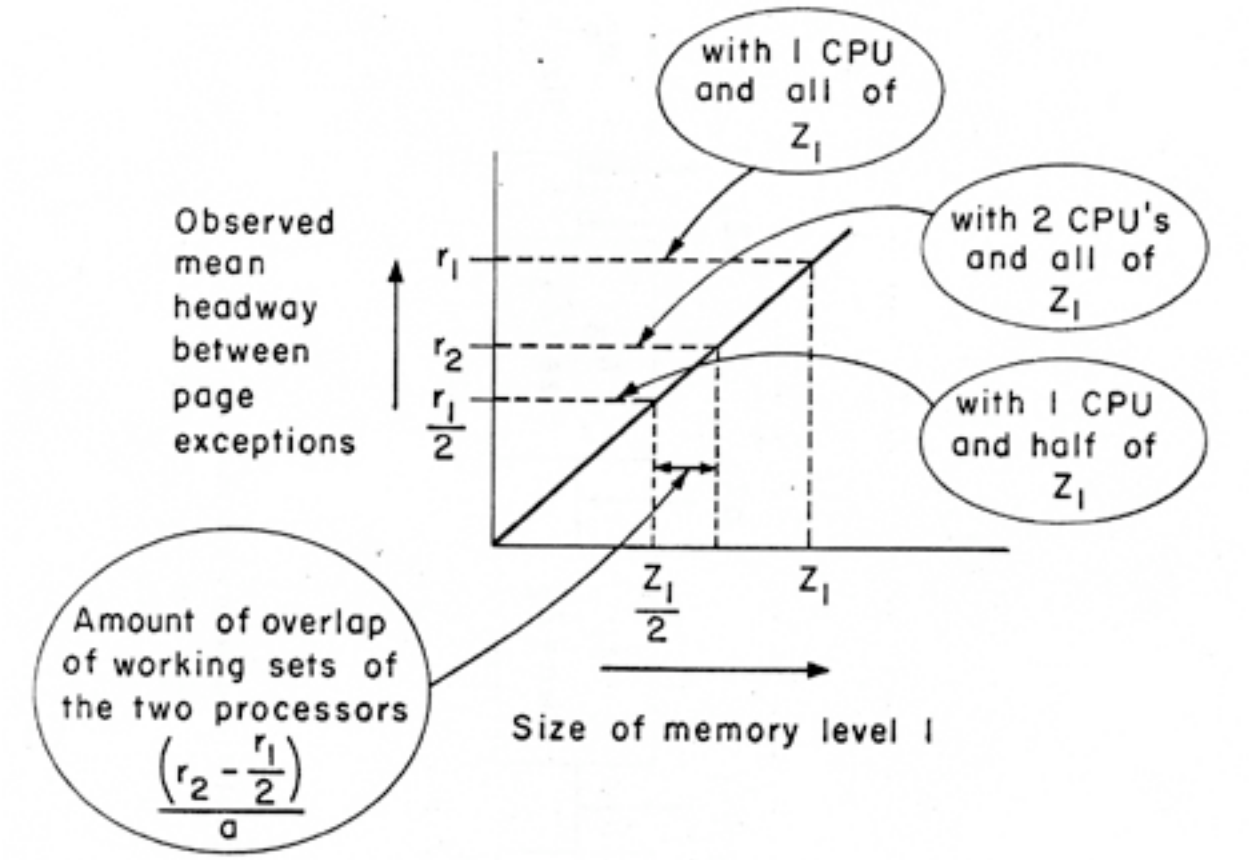

Figure 6: Effective increase in memory size caused by sharing of pages in a 2-CPU system.