UDC 004.4'.416:004.074.3:004.451.3(045)

A. S. Yurchenko, Asst. Prof.,
M. F. Tupitsyn, Asst. Prof.,
I. A. Stepanenko

MINIMIZATION OF THE MEAN ACCESS TIME TO THE HIERARCHICAL MEMORY SYSTEM

Institute of Electronics and Control Systems of NAU, e-mail: iesy@nau.edu.ua

Abstrakt. *Is based on the analytical model, the hierarchical memory of modern computers for solving the problem of choosing the optimal configuration of the memory hierarchy. We analyze the feasibility of the necessary and sufficient conditions for a minimum of the average time to access memory hierarchy.*

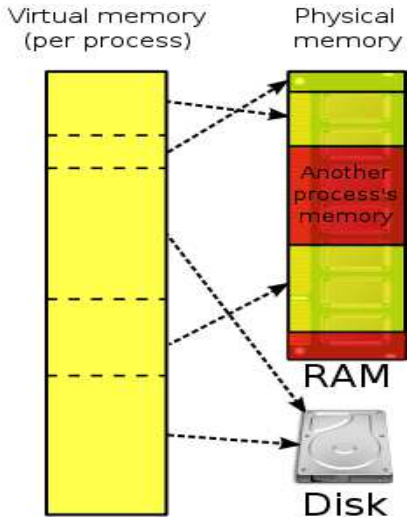
Keywords: modern computer model of the system memory, the average time of treatment, memory allocation.

Introduction and statement of the problem. Architecture of modern computers provides the organization multilevel hierarchical memory, providing access of the CPU to each level of the memory. The productivity of the computer determines the productivity of memory, that is, the average service time of signale memory access. This average service time depends on the size of each level of the memory hierarchy, the page size, the cycle time of memory devices at each level of the hierarchy. Modern computers that operate with large data arrays, require the use of memory devices with different performance and, consequently, with different values. Such a hierarchical memory system is very expensive, and performance evaluations of these systems have become important in the development stage. Over recent years a number of studies appeared that evaluated the effectiveness of multilevel memory systems [1 – 5].

Dynamic memory allocation is closely intertwined with the concept of virtual memory.

Principle of virtual memory suggests that the user in the preparation of its program is not concerned with the physical RAM, actually working in the computer and has some fixed size and the virtual single-level memory, the capacity of which is the whole address space.

In computing, virtual memory is a memory management technique developed for multitasking processing. This technique virtualizes a computer architecture's various forms of computer data storage (such as random-access memory and disk storage), allowing a program to be designed as though there is only one kind of memory, "virtual" memory, which acts like directly addressable read/write memory.



Algorithms moving blocks of information from memory to external memory

Thus, computer must be provided with a sufficient amount of external memory to store all programs that are processed on a computer.

The programmer has the address space in his possession, limited only by the address bus width, regardless of the total amount of system memory, and the memory used by other programs, parallelly processed in a multiprogramming computer.

The redistribution of the total power in the to external memory to make space for the new information is usually one of the following algorithms (figure):

– LRU (least recently used) – most have not been used;
– FIFO – the very old to stay in the RAM;
– Random – random.

Virtual memory, allowing the programmer to handle a very large amount of contiguous address space provided at the disposal of its monopoly, has the usual properties: byte-addressing, access time is comparable with the memory access.

At all stages of the programming, including loading into memory, the program is represented in the virtual address, and only when the virtual machine instruction addresses turn into physical. For each program being run in multitasking mode of its own virtual memory is creates. Each program uses the same virtual address from zero to the largest possible in the given architecture.

To convert virtual addresses to physical, physical and virtual memory is broken into blocks of fixed length, called pages. Amounts of virtual and physical pages are the same. Pages of virtual and physical memory are numbered.

In Denning's works [5] the general estimation and classification of the fragmentation arising at realization of virtual memory are given. The concept "fragmentation" means presence of unused portions in memory which cannot be used. According to [5] there are external, internal and tabulated fragmentations. It spaces ariseng between blocks of memory are not used, the external fragmentation takes place. Formation of "holes" inside of blocks testifies to an internal fragmentation. At virtual memory page (i. e. the fixed length of blocks) the external fragmentation does not happen, as each removed page is replaced another, the same size, and backlashes between them are not formed. However the internal fragmentation can take place; because of the size of demanded memory is approximated in greater party to an integer of pages.

In computing systems with virtual memory and segment distribution all free memory is allocated in the form of set of the free segments scattered between occupied segments as their employment and clearing occurs in the casual manner. D. Knut [6] has proved a rule "fifty percent" which establishes parity between number occupied and number of free segments. According to this rule if the system of memory aspires to a condition of balance at which in system is available on the average N occupied segments the average quantity of free segments is approximately the same.

The degree of an external fragmentation at segment distribution is defined as the total size of free segments. The external fragmentation takes place in case

$$S \rangle \max\{x_i\}, \quad \sum_{i=1}^{m} x_i \rangle S,$$

where S – the needed size of memory, and $\{x_i\}$ (i = 1, 2, …, $m \approx \frac{N}{2}$) – sequence of the sizes of free segments.

The table fragmentation is generated by that the part of physical cells of memory are allocated for the service information of system of dynamic allocation of memory (DAM) - tables of correspondence. Thus, these cells cannot be used for memory allocation.

The internal fragmentation according to [5] takes place only at page virtual memory. Actually this fragmentation is inherent also in virtual memory with segment system of DAM, in particular, because of difficulties accounting and allocation of segments of too small sizes.

Model of the system memory. Let the memory hierarchy consists of various levels M_i, where $1 \le i \le N$. It is assumed that the memory M_1 is the smallest and has the least time to access this memory, and the memory M_N is maximum size and has the maximum time access this memory.

Each level of the memory M_i is characterized [4], by the following parameters: B_i is the cost of memory M_i; b_i is the cost of storing data in the memory unit M_i (for example, the cost of storing one byte); c_i is the size memory M_i in terms of data $B_i = b_i c_i$; t_i is the average time to access data memory unit M_i, rather, the delay calculation, due to access memory M_i; P_i – probability of access memory M_i; $P_i = p_i c_i$.

System characteristics of interest, the total price of the memory hierarchy, and the average time to access this memory hierarchy and the average time to access this memory hierarchy are described by the formulas:

$$T_{avg} = \sum_{i=1}^{N} t_i P_i, \qquad (1)$$

$$S = \sum_{i=1}^{N} B_i. \quad (2)$$

To describe the problem in the design of the memory hierarchy is to minimize the average access time for a given value of S by choosing the parameters of the memory hierarchy. This requires minimization analysis of two functions: 1) the characteristics of the equipment, i. e, the ratio between t_i и b_i; 2) the characteristics of the use, which indicates the distribution of calls to the memory hierarchy, and thus sets the connection P_i і c_i. As in [4], we assume given the number and size of different kinds of memory M_i, as well as the probability Pi of access to memory.

It is known that the unit cost of memory is a decreasing function of the time of treatment [5]. Therefore, the model can be specified by the memory function

$$b_i(t) = b_0 t_i^{-\beta}, \ t_i > 0. \quad (3)$$

For real hierarchical memory systems, parameter values β are in the neighborhood of 0,5 [6]. Using the model memory in the form (3) is consistent with the actual data when the access time of eight orders of magnitude. The values of the parameters b_0 and β may be different depending on the current applications.

Thus, the problem of choosing the optimal configuration of a hierarchical memory system is reduced to minimize the average access time of the memory hierarchy, that is reduced by the optimization of (1) by the constraint (2).

Optimization of memory configuration. To solve the problem of minimizing the average access time T_{avg} restricted by the total cost of the memory system S we use the Lagrange multiplier method. To this end, we construct the Lagrangian

$$F = \sum_{i=1}^{N} c_i p_i t_i + \lambda \left(\sum_{i=1}^{N} c_i b_0 t_i^{-\beta} - S \right).$$

The necessary condition for a minimum average access time T_{avg} is

$$\frac{dF}{dt_i} = c_i p_i + \lambda \left(-c_i b_0 \beta t_i^{-\beta-1} \right) = 0.$$

From this expression, we get:

$$p_i = \lambda b_0 \beta t_i^{-\beta-1}; \quad (4)$$

$$t_i = (\lambda b_0 \beta / p_i)^{1/(\beta+1)}; \quad (5)$$

$$t_i^{-\beta} = (\lambda b_0 \beta / p_i)^{-\beta/(\beta+1)}. \quad (6)$$

Substitute (6) into (2), assuming that the sum of this formula holds with the index j (to avoid confusion in the calculations using the index i):

$$\sum_{j=1}^{N} c_i b_0 (\lambda b_0 \beta / p_j)^{-\beta/(\beta+1)} = S; \quad b_0 (\lambda b_0 \beta)^{-\beta/(\beta+1)} \sum_{j=1}^{N} c_j p_j^{\beta/(\beta+1)} = S; \quad \lambda b_0 \beta = \left(\frac{b_0}{S} \sum_{j=1}^{N} c_j p_j^{\beta/(\beta+1)} \right)^{(\beta+1)/\beta} = S. \quad (7)$$

Substituting (7) into (5) we have

$$t_i^0 = \left(\frac{b_0}{S} \sum_{j=1}^{N} c_j p_j^{\beta/(\beta+1)} \right)^{1/\beta} p_i^{-1/(\beta+1)}. \quad (8)$$

Hence the average time T_{avg} treatment will be described by the expression

$$T_{avg} = \sum_{i=1}^{N} t_i{}^0 P_i = \sum_{i=1}^{N} c_i t_i{}^0 p_i = \sum_{i=1}^{N} c_i \left(\frac{b_0}{S} \sum_{j=1}^{N} c_j p_j{}^{\beta/(\beta+1)} \right)^{1/\beta} p_i^{\beta/(\beta+1)} =$$

$$= \left(\frac{b_0}{S} \sum_{j=1}^{N} c_j p_j{}^{\beta/(\beta+1)} \right)^{1/\beta} \sum_{i=1}^{N} c_i p_i{}^{\beta/(\beta+1)} = S^{-1/\beta} b_0{}^{1/\beta} \left(\sum_{i=1}^{N} c_i p_j{}^{\beta/(\beta+1)} \right)^{(1+\beta)/\beta}.$$

We now show that the solution $t_1{}^0$, …, $t_N{}^0$ satisfies the sufficient condition for a minimum of (1) under the constraints (2), i. e., $d^2F = \sum_{i=1}^{N}\sum_{k=1}^{N} \frac{d^2F}{dt_i dt_k}|_{(t_1^0,\ldots t_N^0)}$, $dt_i dt_k > 0$.

Since $dF / dt_i = c_i p_i - \lambda c_i \beta b_0 t_i^{-\beta-1}$, it is valid $d^2F / dt_i dt_j$ for all $i \neq j$ and $d^2F / dt_i{}^2 = \lambda c_i b_0 \beta(\beta+1) t_i^{-\beta-2}$.

Therefore, the relation is executed $d^2F / \lambda b_0 \beta(\beta+1) \sum_{i=1}^{N} t_i^{-\beta-2} dt_i{}^2 > 0$.

As for the real parameters of hierarchical memory systems p_i, c_i, b_0, β they are positive, it follows from (4) and (8) $\lambda > 0$.

Thus, the formula (8) is the solution of the problem of minimizing the average time to access memory hierarchy T_{avg} to the restriction on the total cost of the memory system S, and hence the solution of the problem for choosing the optimal configuration of a hierarchical memory system.

Conclusion. A mathematical model of hierarchical memory systems is used to solve the problem of minimizing the average time to access memory hierarchy which is restricted by the total cost of the memory system.

References

1. *Morgan Kaufmann.* Publishers. Large and Fast: Exploiting Memory Hierarchy. P. 1–54. March 2013. http://www.cse.msu.edu/~cse420/lectures/Chapter5.pdf.
2. *John Mellor-Crummey, David Whalley, and Ken Kennedy*. 2001. Improving Memory Hierarchy Performance for Irregular Applications Using Data and Computation Reorderings. International Journal of Parallel Programming, Vol. 29, No. 3, P. 217–247. http://cacs.usc.edu/education/cs653/Mellor-MDlayout-IJPP01.pdf.
3. *Rajeev Balasubramonian, David H. Albonesi, Alper Buyuktosunoglu, and Sandhya Dwarkadas*. A Dynamically Tunable Memory Hierarchy. IEEE transactions on computers, Vol. 52, No.10, October 2003. P. 1–16. http://www.cs.utah.edu/~rajeev/pubs/-ieeetoc03.pdf.
4. *Welch T. A.* 1978. Memory hierarchy configuration analysis. IEEE Trans. Comput., Vol. 27, No.5. P. 408–413.
5. *Peter J. Denning*, Virtual Memory, ACM Computing Surveys (CSUR), Vol. 2, No.3. September 1970. P. 153–189.
6. *Knuth D.* The Art of Computer Programming. Fundamental Algorithms, Third Edition (Reading, Massachusetts: Addison-Wesley, 1997), xx+650 p.

О. С. Юрченко, М. Ф. Тупіцин, І. О. Степаненко

Мінімізація середнього часу доступу до ієрархічної системи пам'яті

Запропоновано рішення задачі мінімізації середнього часу доступу до ієрархії системи пам'яті. Виконано аналіз необхідних і достатніх умов такої мінімізації.

А. С. Юрченко, Н. Ф. Тупицин, И. А. Степаненко

Минимизация среднего времени доступа к иерархической системе памяти

Предложено решение задачи минимизации среднего времени доступа к иерархии системе памяти. Выполнен анализ необходимых и достаточных условий такой минимизации.