UDC 689.3.19/16 (045)

[1]A. S. Yurchenko,
[2]A. P. Kozlov

ABOUT WORKING SET FOR DYNAMIC MEMORY ALLOCATION

Aviation Computer-Integrated Complexes Department, National Aviation University, Kyiv, Ukraine
E-mails: [1]ayurchenko@yahoo.com, [2]ap_kozlov@ukr.net

Abstract—*A New approach to dynamic allocation of memory paging is described. The approach is distinguished for a choice of the dynamic allocation of memory paging algorithm parameters during a computation process for achieving maximum loading of the central processor.*

Index Terms—Dynamic paging memory allocation; working set; model for program behavior; fragmentation; memory simulation.

I. INTRODUCTION

Memory management is the act of managing computer memory at the system level. The essential requirement of memory management is to provide ways to dynamically allocating portions of memory to programs at their request, and free it for reuse when it is no longer needed. This is critical to any advanced computer system where more than a single process might be underway at any time [1].

Several methods have been devised that the increasing the effectiveness of memory management. Virtual memory systems separate the memory addresses used by a process from actual physical addresses, allowing separation of processes and increasing the effectively available amount of RAM using paging or swapping to secondary storage. The quality of the virtual memory manager can have an extensive effect on overall system performance [1].

II. A DYNAMIC MEMORY ALLOCATION ALGORITHM

For the last years most of the articles about dynamic paging memory allocation (DPMA) have been described algorithms, which contrasts with a "working set" WS algorithm represented by Denning [2]. From the programmer's standpoint, the working set of information is the smallest collection of information that must be presented in random access memory (RAM) to assure efficient execution of the program, without interrupting because of lack of pages.

There are main working set algorithm's actions: to remove a page from the working set, if this page is not referenced within the preceding r (working set parameter) time units, to include a page in the working set, if this page is referenced, but it is not in RAM. By the working set principle, program can use central processing unit (CPU), when program's working set is in RAM and pages cannot be removed from RAM.

There are advantages of using this algorithm: variable size of RAM, removal from working set useless pages of this program. Disadvantages: hard to choose appropriate value of parameter r, which must be permanent, lack of control algorithm work to assure efficient execution of the program.

That parameter r must be changed, showed in [2], where was introduced a modified working set paging algorithm, that has functions as following.

1. Any page, which was not referenced within the preceding r time unites to, is removed from the main memory.
2. At the time of a page fault, if the time since the last reference is greater than $K \cdot r$ ($0 < K \leq 1$), then the new page is replaced to the least recently used page, otherwise there the space allocation increases by one page frame.

This algorithm was introduced to remedy one of the defects of the previous algorithm, peak expansion of working set, which caused by sharp changes in program locality (for example during alternation of translator phases), old locality pages is not removed, when new locality page is entered.

As opposed to the working set algorithm, Chu and Opderbeck introduced the algorithm PFF, which uses the measured page fault frequency as the basic parameter for the memory allocation decision process. This algorithm try to control itself actions. In general, a high page fault frequency indicates that a process is running inefficiently because it is short of page frames. A low page fault frequency, on the other hand, indicates that further the increase in the number of allocated page frames will not considerably improve the efficiency and, in fact, might result in waste of memory space. Therefore, to improve system performance one or more pages frames could be freed.

The basic policy of the PFF algorithm is: whenever the page fault frequency rises above a given critical page fault frequency level P, all referenced pages which were not in the main memory – therefore causing page fault – are brought into the main memory without replacing any pages.

In [3] – [5] it is affirmed that as the result of modeling this algorithm it is better than the working set algorithm. Work [6], where are represented results of modeling algorithms LRU, WS and PFF and investigated one of the principle of optimal multiprogramming [7], is contradicted to the previous conclusion. 8 types of program are showed that for effective WS algorithm work, different r values (the same with parameter P for PFF algorithm). For effective work for both of those programs, is necessary less different r value for WS algorithm, than different P value for PFF algorithm. The conclusion is: WS better than PFF, but one question is still opened. How does to choose those different r and P values?

This work will try to give the answer for that question. Here is offered algorithm that use all benefits of WS algorithm, but does not have disadvantages that was discussed in the literature (time system costs, problem with choosing critical r parameter), disadvantage that was represented in this work (lack of control algorithm work to assure efficient execution of the program).

The main efficient execution of the DPMA system must not be the measured page fault frequency or using RAM, it must be the coefficient of using CPU (K_{CPU}). Dynamic paging memory allocation with this efficient execution can easy detect and avoid "thrashing", because K_{CPU} can tell about computer much more than, for example the page fault frequency, which control PFF algorithm work.

The offered algorithm (AI) can change parameter r for maximum K_{CPU} value. Figure 1 shows the flowchart that represents algorithm for determining parameter, where $r_{\min}$, $r_{\max}$, r_{act}, r_{des} are minimum, maximum, actual and desired value of the parameter r, K_{CPU} and K_{CPU1} are actual and desired value of the coefficient of using CPU. Calculative process is performed to determine K_{CPU} value at the extended time T ($T>>r$). Algorithm can work in one of two regimes: searching (STEP = 0) and stable (STEP = 1). Algorithm uses searching regime to determine K_{CPU} value for different r (to find next value of r, previous value multiply by f) and to choose value of r maximum K_{CPU}. Selected value of r is desired for stable regime for this algorithm. As such, this algorithm can be investigated for DPMA.

III. Model for Program Behavior

To get the results of DPMA modeling, it is necessary to work out two models: model for program behavior and model for DPMA system. The first describes system resource requirements of executed program and the second divides those resources between different programs. In this research, models are based on the models that described in [8] – [10].

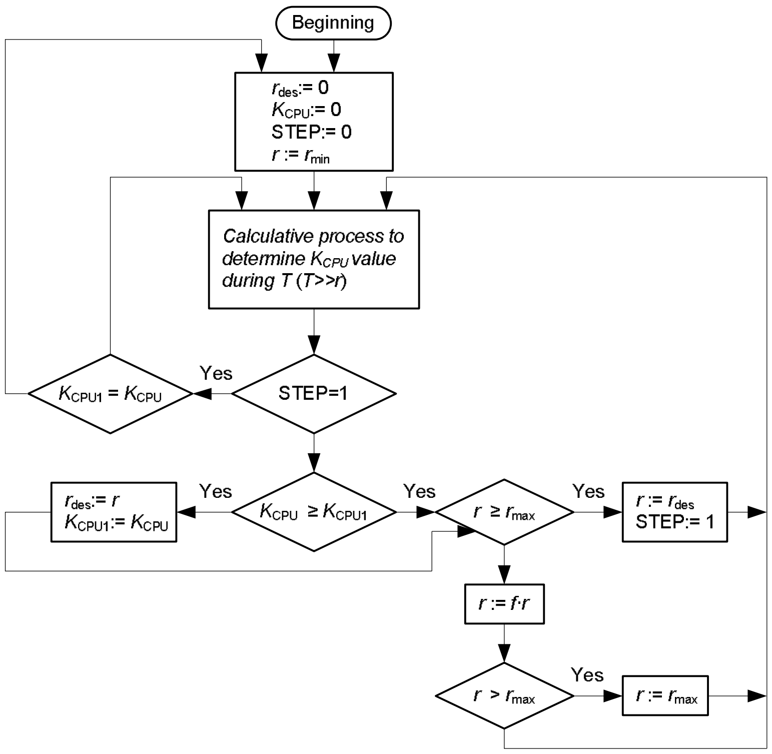

Fig. 1. The determination of parameter r

During the DPMA algorithm researching several different type of program were modeling. There are just two of them. All ith programs can be determined by using some page numbers N_i, full time T_i, time interval inside T_i, when some page must be in RAM, thus program can work. N_i and T_i can be determined by using uniform distribution with parameters $(2, J_1)$, (B_3, B_4). First page of all programs is resident and must be in RAM during full time T_i.

Different types of programs have different way to create pages (live time) intervals. Assumed that locality is so important for programs, it means that during one time interval T_i for solving a program it is necessary to have one page group of that program in RAM, during other time interval other page group of that program.

First program type can be described as following: every jth page of any ith program can be determined by using time intervals t_{ij1} and t_{ij2} $(t_{ij1} + t_{ij2} = T_i)$, t_{ij1} can be found by using uniform distribution with parameters $(0, a{\cdot}T_i)$ for some $a(0 < a < 1)$, and $t_{ij2} = T_i - t_{ij1}$.

For any program page with probability y interval t_{ij1} is time interval, when page is necessary (presence interval), and during second time interval t_{ij2} that page isn't necessary (absence interval). With probability $1 - y$ interval t_{ij1} for this page is absence page interval, and interval t_{ij2} – presence interval.

Second program type can be described as following: every jth page of any i-th program can be determined by using three time intervals t_{ij1}, t_{ij2} and t_{ij3} $(t_{ij1} + t_{ij2} + t_{ij3} = T_i)$, t_{ij1} and t_{ij2} can be found by using uniform distribution with parameters $(0, a{\cdot}T_i)$, $(0, b{\cdot}T_i)$ for some $a(0 < a < 1)$ and $b(0 < b < 1)$, and $t_{ij3} = T_i - t_{ij1} - t_{ij2}$. Further for any program page with probability y intervals t_{ij1} and t_{ij3} are presence intervals of that page in RAM (t_{ij2} is the absence interval), with possibility $1 - y$ those time intervals are absence intervals (t_{ij2} is the presence interval).

IV. About Modeling Program

In this work DPMA algorithms are investigated by using designed modeling program, which allows imitate implementation of different program types. Programs behaviors are described earlier.

We assumed that at the beginning all pages of any program are in secondary memory (disk storage). Resident page must be moved from secondary memory to RAM to start the program processing. It is necessary to create input descriptor of desired page in RAM. Time that is necessary for moving the page from secondary memory to RAM is equal to waiting time of corresponding descriptor in input/output turn and immediate time of page moving.

After inputing the resident page, program stays in turn of programs that are ready to be used in CPU. This turn is called- turn of active programs. To each program which wait in that turn is given quantum time to be processed in CPU. During that quantum time program can address to the page fault in RAM, process of the program can be finished or given quantum time exhausted before the end of program processing. In the first case program is deleted from the turn of active programs and addressing to the procedure of RAM occurs for deserved page of given program. If memory for deserved page can be provided, then input descriptor will be created of deserved page from the disk to RAM. When deserved page is inputted into RAM, given program will stay at the end of the turn of active programs.

If in the result of addressing the fault page it is no place for in RAM, then we remove all pages of that program from RAM to secondary memory. When all pages are removed to secondary memory, then that program will stay in the turn of discarded programs. Discarded programs can become active again, if there is a possibility to input all pages of that program, which were present in the RAM before removing that program to secondary memory. The processing of new program will start in a condition when there is no queue of discarded programs and when there is enough space for resident page of that program in RAM. The end of the program occurs during the program processing. In that case, all pages of given program, which are sated in RAM are removed to secondary memory. When all pages are removed, program processing considered as-done. If quantum time ends before the end of the program processing, then program will be removed from the beginning to the end of the queue of active programs.

The modeling programs, and DPMA algorithms, were written in high-level language (C++). The activation procedure of received and discarded programs, the formation of input/output descriptors, memory provision and deallocation were written in microinstruction language, where the runtime for microinstruction is known. After those procedures were written in two languages branches of facilitation programs in C++ then they were calibrated in time by using appropriate programs that were written in microinstruction language, because for such programs it is possible to determine right time of different branches processing.

After calibration, modeling runs in C++ for which it was chosen continuous - discrete systems. As microinstructions it was chosen language of developed computer. The time to execute microinstructions like: register transfer, reading from RAM to local memory, CPU respectively equal $16{\times}10^{-8}$ and $112{\times}10^{-8}$ C. Distributed memory consists of 32-bits words, and memory is provided with accuracy to the word.

The assessment of modeling results of proposed DPMA algorithm is done by the using of an utilization ratio CPU, that can be determined as follows:

$$K_{\text{CPU}} = ((T - T_{\text{down}} - T_{\text{sys. out}})/T)\cdot 100\%,$$

where T is the long time interval; T_{down} is the CPU downtime and $T_{\text{sys. out}}$ is the system delay during time T.

V. Modeling Results

The main goal of modeling – is to show, how proposed algorithm AI works. It is based on Denning's algorithm, but at the same time has changeable value of r step. As the result of searching, algorithm choses any r at which K_{CPU} is maximum and only when r is already found, the further processing of program will be performed. Below, the results modeling results for these cases are shown, for example when algorithm has following parameters: $r_{\min} = 0.01$, $r_{\max} = 10.0$, $f = 10.0$, $T = 120.0$. The results of modeling were determined as follows: values of CPU utilization ratio were determined during researching the r parameter by algorithm AI for different r, represented in Table I.

Table I

Results of simulation

τ	First type of program			Second type of program	
	$\alpha = 0.1$ $\gamma = 0.99$ $B_4 = 2.1$	$\alpha = 0.1$ $\gamma = 0.5$ $B_4 = 2.1$	$\alpha = 0.4$ $\gamma = 0.99$ $B_4 = 1.0$	$\alpha = 0.4$ $\beta = 0.08$ $\gamma = 0.99$ $B_4 = 2.1$	$\alpha = 0.01$ $\beta = 0.99$ $\gamma = 0.5$ $B_4 = 4.2$
0.01	51.15	43.62	33.4	27.88	63.33
0.1	68.08	58.05	35.8	37.04	65.06
1.0	49.87	53.10	31.1	50.90	72.12
10.0	48.81	49.96	26.9	48.56	67.01

For first type programs (2) $T_i \approx 1.05$, $t_{ij1} \approx 0.05$, $t_{ij2} \approx 1.0$, and the optimal value $r_{\text{opt}} = 0.1$ was chosen by AI algorithm, $K_{\text{CPU}} = 68.08\%$. In (3) it is represented modeling results of first type program, but with $y = 0.5$, not a 0.99. Optimal $r_{\text{opt}} = 0.1$ and $K_{\text{CPU}} = 58.06$. Results of comparison between (2) and (3) shows that in both cases optimal value $r_{\text{opt}} = 0.1$ is less than full time of program processing. It means that during program processing it dedicates first local program page, which removes from RAM, before the end of program processing, to exempt space for other programs. It is easy to see that in the first case difference between maximum and minimum K_{CPU} values is bigger than in the second one. It occurs because in the first case the first program locality occupies 99% of all memory, which is needed for the program (in the second case 50%). Therefore, during the transition to the second locality, when first one hasn't removed yet, it is occupied more memory than in the second case. That is why here is the difference between maximum and minimum K_{CPU} value.

The parameters, which have the same values for these types of programs: $B_2 = 0.0$; $J_1 = 20$; size of page = 256.

Let us consider the modeling results, given in (4) for first type program. In this case: $t_{ij1} \approx 0.1$ and $t_{ij2} \approx 0.4$. Optimal value – $r_{\text{opt}} = 0.1$ was found by AI algorithm, $K_{\text{CPU}} = 35.8\%$. These results shows that difference between maximum (first program page locality can be detected) and minimum (cannot be found) is equal to $K_{\text{CPU}} = 9\%$. This difference is quite small because $t_{ij2} \approx 0.4$ (4) < $t_{ij2} \approx 1.0$ (2).

The modeling results of algorithm AI works during second type processing are represented in (5) and (6). There are represented programs with small value of t_{ij2}($t_{ij2} \approx 0.04$) (type A) in (5) and programs with high value t_{ij2}($t_{ij2} \approx 1.0$) (type B) in (6), where t_{ij2} – time interval, when programs don't need the first locality pages. That interval is set to be between two time intervals t_{ij1} and t_{ij3} during which program needs pages of the same locality.

For type A programs algorithm AI determines $r_{\text{opt}} = 1.0$ and $K_{\text{CPU}} = 50.90\%$. r_{opt} value much bigger t_{ij2}($t_{ij2} \approx 0.04$), because after program processing during t_{ij1}($t_{ij1} \approx 0.2$) first page locality doesn't remove from RAM, because after time t_{ij2} its needed again . That solution is better than removal of the first page locality and imputing them back to RAM.

For type B program (6) when time t_{ij1} expired, the first program page locality is better to be removed, and after inputted again. Such a conclusion can be drawn from the condition when $r_{\text{opt}} < t_{ij2}$ so the first page locality removes, otherwise that locality doesn't remove and coefficient K_{CPU} decreases in comparison with K_{CPU} at r_{opt}. In this way, modeling results shows that algorithm AI provides searching for parameter r at which value of CPU utilization ratio is maximum.

VI. Conclution

In this work new approach is described for the DPMA. The main feature of that approach is possibility to choose DPMA algorithm parameters during computational process with the goal to achieve maximum CPU utilization.

The given approach was investigated also to choose the optimal level of multiprogramming. It is known, that the main purpose of multiprogramming is to achieve full utilization of different computer devices, firstly CPU. The sense of this approach is to find level of multiprogramming at maximum CPU utilization ratio by measuring level of multiprogramming. So during further program processing the founded value will be used like optimal, until the input stream of processing programs will require to search for the new value of multiprogramming level.

REFERENCES

[1] Donald Knuth, *Fundamental Algorithms*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89683-4. Section 2.5: Dynamic Storage Allocation, pp. 435–456.

[2] P. J. Denning, "Working sets part and present." *IEEE Trans.* Software Eng., 1980, 6, no. 1, pp. 64 – 84.

[3] A. J. Smith, "A modified working at paging algorithm." *IEEE Trans. Comput.*, 1976, 25, no. 9, pp. 907–914.

[4] W. W. Chu, and H. Opderbeck, "Performance of the page fault frequency replacement algorithm in multiprogramming environment." *In: IFIP Congress–74.* Stockholm, 1974, pp. 235–241.

[5] W. W. Chu, and H. Opderbeck, "The page fault frequency replacement algorithm." *In: Proc. AFIPS 1972. Fall joint computer conf.*, 1972, no. 41. pp. 597–609.

[6] W. W. Chu, and H. Opderbeck, "Performance of replacement algorithm with different page sizes." *Computer*, 1974, 7, no.11, pp. 14–21.

[7] G. S. Graham, and P. J. Denning, "On relative controllability of memory policies." *In: Computer performance: Proc. of the inert. symp. on comput. performance modeling, measurement and evaluation.* New York, 1977, pp. 411–428.

[8] A New approach to minimize page fault. 2012 International Conference on Information and Computer Networks (ICICN 2012) IPCSIT vol. 27 (2012), http://www.ipcsit.com/vol27/20-ICICN2012-N048.pdf

[9] Hidden costs of memory allocation. https://randomascii.wordpress.com/2014/12/10/hidden-costs-of-memory-allocation/

[10] Memory profiling for application performance. https://blogs.windows.com/buildingapps/2012/01/31/memory-profiling-for-application-performance/.

Received January 04, 2016

Yurchenko Alexander. Candidate of Engineering. Assistant professor.
Aviation Computer-Integrated Complexes Department, National Aviation University, Kyiv, Ukraine.
Education: Moscow Phisics-thechnical Institute, Moscow, Russia (1975).
Research area: operating system, dynamic allocation memory.
Publication: 56.
E-mail: ayurchenko@yahoo.com

Kozlov Anatoliy Pavlovich. Candidate of Engineering. Assosiate Professor.
Aviation Computer-Integrated Complexes Department, National Aviation University, Kyiv, Ukraine.
Education: Kiev State University named T. G. Shevchenko, Kyiv, Ukraine (1965).
Research interests: Capacitive transducers with non-uniform electromagnetic field. Capacitive meters of parameters small altitude of the flight aircraft. The use of capacitive transducers in automatic control small-altitude of the flight aircraft.. Publications: 48.
E-mail: ap_kozlov@ukr.net

О. С. Юрченко, А. П. Козлов. Про робочий набір для динамічного розподілу пам'яті

Описано дослідження нового підходу для динамічного розподілу сторінкової пам'яті. Особливістю цього підходу є вибір параметрів алгоритмів протягом обчислювального процесу з метою досягнення максимального завантаження центрального процесора.

Ключові слова: динамічний розподіл сторінкової пам'яті; робочий набір; модель поведінки програми; фрагментація; моделювання розподілу пам'яті.

Юрченко Олександр Сергійович. Кандидат технічних наук. Доцент.
Кафедра авіаційних комп'ютерно-інтегрованих комплексів, Національний авіаційний університет, Київ, Україна.
Освіта: Московський фізико-технічний інститут, Москва, Росія (1975).
Напрям наукової діяльності: операційні системи, динамічний розподіл пам'яті.
Кількість публікацій: 56.
E-mail: ayurchenko@yahoo.com

Козлов Анатолій Павлович . Кандидат технічних наук . Доцент.
Кафедра комп'ютерно-інтегрованіх комплексів, Національний авіаційний університет, Київ, Україна.
Освіта: Київський державний університет имени Т. Г. Шевченка, Київ, Україна (1965).
Напрям наукових інтересів: Ємнісні перетворювачі з неоднорідім електромагнітнім полем. Ємнісні прилади вімірювання геометричних параметрів мало висотного польоту повітряного судна. Використання ємнісніх перетворювачів в системах автоматичного управління мало висотним польотом повітряного судна.
Публікації : 48 .
E-mail: ap_kozlov@ukr.net

А. С. Юрченко, А. П. Козлов. О рабочем наборе для динамического распределения памяти
Описано исследование нового подхода к динамическому распределению страничной памяти. Особенностью этого подхода является выбор параметров алгоритмов в течении вычислительного процесса с целью достижения максимальной загрузки центрального процессора.
Ключевые слова: динамическое распределение страничной памяти; рабочий набор; модель поведения программы; фрагментация; моделирование распределения памяти.

Юрченко Александр Сергеевич. Кандидат технических наук. Доцент.
Кафедра авиационных компьютерно-интегрированных комплексов, Национальный авиационный университет, Киев, Украина.
Образование: Московский физико-технический институт, Москва, Россия (1975).
Направление научной деятельности: операционные системы, динамическое распределение памяти.
Количество публикаций:56.
E-mail: ayurchenko@yahoo.com

Козлов Анатолий Павлович. Кандидат технических наук. Доцент.
Кафедра компьютерно-интегрированых комплексов, Национальный авиационный университет, Киев, Украина
Образование: Киевский государственный университет имени Т. Г. Шевченко, Киев, Украина (1965).
Область научных интересов: Емкостные преобразователи с неоднородным электромагнитным полем. Емкостные устройства измерения геометрических параметров мало высотного полета воздушного судна. Использование емкостных преобразователей в системах автоматического управления мало высотным полетом воздушного судна.
Публикации: 48
E-mail: ap_kozlov@ukr.net