UDC 004.051 : 004.451.35 : 519.683.4

D. N. Samoilenko, PhD

MEMORY TRACING INFLUENCE ON ALGORITHM COMPLEXITY

Experimental measurements are carried out for the working time of programs that make different steps in the operational memory. Essential increase (up to10 times) of working time was found out for certain steps. It was shown that usage of static memory allocation may cause the 26 % advantage in the time for reading and writing operations.

Keywords: *memory tracing, algorithm complexity.*

Д. Н. Самойленко, канд. техн. наук

ВЛИЯНИЕ ОПЕРАЦИЙ С ПАМЯТЬЮ НА СЛОЖНОСТЬ АЛГОРИТМОВ

Проведены эксперименты по измерению времени работы программ, реализующих шаги различной величины в оперативной памяти. Обнаружены существенные увеличения (до 10 раз) времени работы программ для некоторых величин шага. Установлено, что использование статически выделяемой памяти приводит к выигрышу до 26 % во времени выполнения операций чтения и записи.

Ключевые слова: *операции с памятью, сложность алгоритмов.*

Д. М. Самойленко, канд. техн. наук

ВПЛИВ ОПЕРАЦІЙ З ПАМ'ЯТТЮ НА СКЛАДНІСТЬ АЛГОРИТМІВ

Проведені експерименти щодо вимірювання часу роботи програм, які реалізують кроки різної величини у оперативній пам'яті. Помічені суттєві зростання (до 10 разів) часу роботи програм для окремих розмірів кроків. Встановлено, що використання статично резервованої пам'яті надає перевагу до 26 % у часі виконання операцій читання та запису.

Ключові слова: *операції з пам'яттю, складність алгоритмів.*

Introduction

Dynamic usage of memory is the most popular method for volumetric data processing and long arrays allocation. With long arrays dynamic databases, experimental data in real time mode, matrixes and vectors in multidimensional spaces could be computed. Memory sizes grow together with the society evolution so researches of long arrays remain topical.

Term long means that array greater than maximum possible in DOS mode. Tracing of such memory is possible only in processor safe mode. In this mode granularity of memory and page principle could involve time variations for array samplings.

Experiment

To discover time aspects of arrays sampling lets form a goal to measure the time of every N-th element processing in a given array. Studied processes were reading and writing operations.

Arrays were created static as well as dynamic for time comparison in dependence from creation methods.

For programming C++ 5,02 by Borland International was used. Time was measured with help of the recommended (in Borland documentation) method:


```
clock_t start, end;
start = clock();

Fragment being measured

end = clock();
printf("The time was: %f\n", (end - start) /
CLK_TCK);
```

The object of research was a linear array with 1 000 000 elements of double type. The task was measuring a time of operations with equidistant array elements - every second, every third, etc. Similar operations are typical for databases and frame models of knowledge bases in which appeal to the next element corresponds to shift to the next track or frame.

For simplicity of comparison, each operation was performed a fixed number (1000) of times. To improve the accuracy of time measurement experiment was repeated. Criteria for selecting the number of repetitions performed resulting time for the expected value of which was elected 0,1-1 sec. For a PC that performed the experiment (CPU AMD Duron 1.4 GHz, RAM 512 Mb) the optimal number of repetitions was 10 000 and did not change for different fragments being measured.

Common to all experiments the part of source code looks like:

```
1.  int N,Nmin=1,Nmax=300;
2.  double T;
3.  double far *px;
4.  clock_t start, end;
5.  int i, j;
6.  for(N=Nmin;N<=Nmax;N+=1)
7.  {   start=clock();
8.      for(i=0;i<10000;i++)
9.      {initialization
10.       for(j=0; j<1000; j++)
11.       {N*j-th element processing}
12.     } end=clock();
13.     T=(end - start) / CLK_TCK;
14.     sprintf(str,"%d\t%02.3f",N,T);
15.     fprintf(f,"%s\n",str);
16. }
```

The lines in the program means:

1. loop variable and its ranges
2. time variable
3. array pointer
4. variables for clock scanning
5. loop variables
6. main loop
7. start time by clock
8. repetition for accuracy increasing
9. variables restoring
10. the body of experiment – equidistant
11. elements processing
12. finish time by clock
13. working time
14. preparing data for output
15. save data in file, assigned to “f”
16. repeat for other N

For array creation variable RESERV was used. In the case of dynamic creation the variable was initialized as

double *RESERV=new double[LENGTH];

in the case of static creation as

double RESERV [LENGTH].

Described in the previous code fragment pointer px was used to array access. RESERV variable was used as a constant for array beginning identification (restoring).

Writing operation implemented by saving zero constant into the given element of array. For additional comparison of element accessing mode operations were formed in two configurations: directly through dereferencing of shifted pointer (*px = 0), and in the static way with a given displacement (px[n] = 0).

Read operations implemented by using of the same elements in addition operation (*px + *px or px[n] + px[n]). The data from the corresponding array cell should be loaded into the PC registers, so it would be read. The result of summation was not preserved for elimination of additional write operations (assignment).

Time measurement results were saved in an external file and processed in the Origin package (by Origin Lab).

Results and discussion

Results of timing for all operations and all modifications of code had almost the same behavior. Fig. 1 shows the results regarding work time of writing operation with pointer dereferencing (*px = 0). Results of other programs timing differ in absolute time value, but retain the relative position and intensity of local extreme and show similarity with Fig. 1.

The dependence presented in Fig. 1, shows some features. First, there are extremes of working time for specific values of N. Second, it could be observed the rising of graph bottom edge with increasing N.

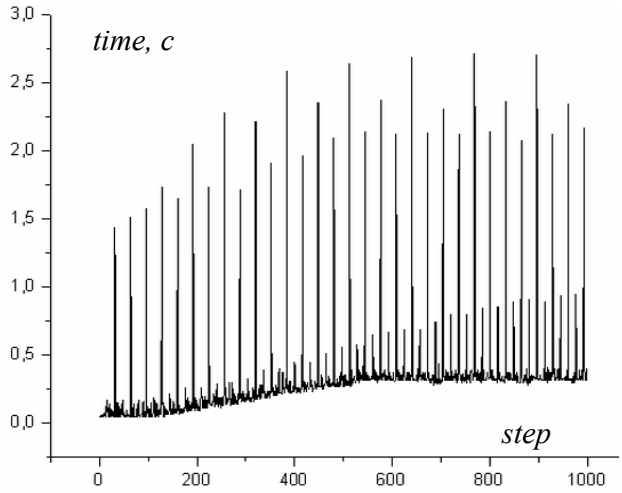

Fig. 1. Memory tracing time (t) dependence from step dimension (N)

The most obvious reason of extremes is that service starts in a multitask system. But this reason was simply refuted by program restarts. The extremes stead on the same places with the same intensity. The same position and the similar intensity show extremes for different codes. So, the nature of extremes could not be explained by activation of system programs.

It could be calculated that distance between adjacent extremes is constant and equal $\Delta N=32$. As far as variable of double type use 8 bytes of

memory, the distance ΔN corresponds to 8*32=256 bytes.

In such jumps increasing of time can be explained by two factors. The first part of time increasing can be associated with the jump to a new row in memory chips and, consequently, the necessity of row access signal RAS [5,6], which duration may be 2-5 times longer than the CAS signal [7, Fig. 7.4], generated for every jump.

The second component is related to increasing of frequency of accessing cells with shift of 64 kB (256 * 256 = 64 kB) in the virtual address space that does not appear in physical memory to the control system software errors [8, pp.890-891]. Time for such requests should increase.

The second feature allows us to locate two parts with different nature of the minimum work time (bottom of chart) in the dependence shown on Fig.1. The first part corresponds to interval from N = 1 to N = 512 shows growth of minimal time, the second – from N = 513 to N = 999 is related to constant value. This feature can be explained in terms of "page" addressing principle in a processor protected mode. Memory page size is 4 kb [8, p.890], and array access operations could occur as within a single page, as in different pages.

The first part of graphics could be associated with different combinations of operations with cells that are on the same and on different pages. With increasing N increases the relative number of jumps to different pages, so the work time increases. In the break point N = 512 (512 * 8 = 4 kB) the distance between neighboring cells in array become greater than a page. In other words, adjacent accessed cells are in different memory pages. In that case all operations run in different pages, so the time of such operations has a maximum value.

The presence of program working time local extremes justifies the additional researches especially for developing applications that use large amounts of data. At best gains in time of 5-10 times could be achieved by making adjustments to the array tracing algorithm or using arrays with optimal length.

The second problem for the research was a comparison of operations in different forms and arrays constructed either statically or dynamically. It should be noted that significant differences in the time of access operations in different forms (means access by * or by [ ]) was not observed. All differences were in the accuracy limits.

At the same time, operations with static and dynamic arrays differ in time much more essentially. Writing time for static array was 0,315 s while for dynamic – 0,361 s. (for the same code). Reading time (for the previous code) was, relatively, 0,238 s. and 0,291 s. We could see, that difference corresponds to 16 % and 26 % for different operations.

The difference in average time for static and dynamic arrays contradict to given in [3] difference in 3 times. It is especially actual because in [3] exactly experimental time was compared. The difference may be explained by using different programming languages or comparison operations for the different conditions (one value of the extremum, otherwise - no). In any case, these differences justify the necessity for further studies of similar phenomena for different programming languages, different operational environments and devices with different structure of memory chips, including PC computers, mobile devices and microprocessor control systems.

Conclusions

The runtime of a program that uses long array tracing strongly depends on the array tracing algorithm. In the worst case for specific relative displacement the runtime increases in 5-10 times. The cause of the time growth is considered in the principles of memory chips producing and the structure of virtual address space.

It was shown that using of static organized arrays in comparison with dynamic, has 16-26 % more efficient running time of the reading and writing operations.

Prospects for further research are seen in conducting similar experiments in different programming languages, for different operational environments and devices with different structure of memory chips.

References

1. Зачесов Ю. Программирование алгоритмов, требующих больших объемов оперативной памяти / Ю. Зачесов. КомпьютерПресс, 2000. – № 12.

2. Агратин Е. Г. Информационное обеспечение пользователей. Научно-практическая конференция "Проблемы обработки больших масивов неструктурированных текстових документов"/ Е. Г. Агратин [WWW документ]. URL http://www.fep.ru/text/dataarrays07.html

3. Волосенков В. Массивы. Статические или динамические? (n.d./2008) [WWW документ]. URL http://www.realcoding.net/article/view/1750

4. Borland C++ User's Guide // електронна документація, інтегрована у пакет "Borland C++ 5.02 by Borland International". – clock, clock example.

5. Энциклопедия современной памяти / С. Пахомов //КомпьютерПресс, 2006. – №10.

6. Учебное пособие АППАРАТНЫЕ СРЕДСТВА ПК. [WWW документ]. URL http://www.msclub.ce.cctpu.edu.ru/bibl/ASVT/asvt3.htm.

7. Шнитман В. Принципы организации основной памяти в современных компьютерах / В. Шнитман // Информационно аналити-ческие материалы Центра Информационных Технологий. [WWW документ]. URL http://citforum.amursu.ru/hardware/svk/glava_8_1.shtml

8. Современные операционные системы. 2-е изд. / Э. С. Таненбаум – СПб.:Питер, 2002. – 1040 с.

Received 31.10.2011

References

1. Yuri Zachesov. Programming algorithms that require large amounts of RAM. ComputerPress, 2000. – № 12 [in Russian].

2. Agratin Ye.G. Information support for users. Scientific-practical conference "Problems of handling large ranges cover unstructured text documents." [WWW document]. URL http://www.fep.ru/text/dataarrays07.html [in Russian].

3. Vladimir Volosenkov. Arrays. Static or dynamic? (n.d./2008) [WWW document]. URL http://www.realcoding.net/article/view/1750 [in Russian].

4. Borland C + + User's Guide // E-dokumentatsiya, integrovana in package "Borland C + + 5.02 by Borland International". – Clock, clock example [in Russian].

5. Sergey Pakhomov. Encyclopedia of modern memory. ComputerPress, 2006. – № 10 [in Russian].

6. Textbook HARDWARE PC. [WWW document] [in Russian]. URL http://www.msclub.ce.cctpu.edu.ru/bibl/ASVT/asvt3.htm.

7. Shnitman B. Principles of organization of main memory in modern computers. / / Information-Analytical Center of Information Technologies of the materials. [WWW document] [in Russian]. URL http://citforum.amursu.ru/hardware/svk/glava_8_1.shtml.

8. Tanenbaum E. S. Modern Operating Systems. 2nd ed. St. Petersburg: Peter, 2002. – 1040 p. [in Russian].

Denis N. Samoilenko,
PhD, vice-director for educational work
European university
Mykolaiv affiliate.
Morehidna 2A, Mykolaiv, 54010, Ukraine,
e-mail: DenNikSam@gmail.com
t.:+38 (0512) 44-06-37