Runtime Customization of Swap Partition to Improve the Performance of a Number Crunching Algorithm

Rajeev Raghuvanshi
Student of ME
Dept. of Information Technology
Medi-Caps Institute of Technology and Management, Indore (M.P.)

Rakesh Verma
Assistant Professor
Dept. of Computer Science
Medi-Caps Institute of Technology and Management, Indore (M.P.)

Dharmendra Sharma
Associate Professor
Dept. of Computer Science
Sushila Devi Bansal College of Engineering, Indore (M.P.)

ABSTRACT
This paper focuses on the enhancement of the practical performance of an algorithm and data structures. It provides a way to achieve the effect on performance parameters of operating system. The proposed work has been done about the performance of a number crunching program (Matrix Multiplication) by varying the swap area at run time. We feed a C program as input to the system that executes it for different value of swap area and records the performance parameters as real time, system time, CPU utilization, page faults occurs during execution, context switching due to time slice and context switching due to input output etc. It also presents the way of enhancement in performance of program by customizing the swap partition of operating system.

General Terms
Algorithm, Performance Parameter, Operating System, Matrix Multiplication etc.

Keywords
MIPF, CSIO, Swap area CST, CSIO, PSO, System Time, User Time, Page Fault, Cache Size.

1. INTRODUCTION
An algorithm is a sequence of finite instructions, a well defined computational procedure that takes some value or set of values, as input and produces some value, or set of values, as output [1]. In mathematics, computing and related subjects, an algorithm is a step by step process, often used for calculation and data processing [2]. It is formally a type of effective method in which a list of well-defined instruction for completing a task will, when given an initial state, proceed through a well-defined series of successive states, eventually terminating in an end-state. The efficiency of an algorithm can be decided by measuring the performance of an algorithm. We can measure the performance of an algorithm by computing two factors [3].

A. Amount of time required by algorithm to execute.
B. Amount of storage required by an algorithm.

This is popularly known as time complexity and space complexity of an algorithm.

1.1 Space Complexity
The Space complexity can be defined as amount of memory required by an algorithm to run. Space complexity is also measured with Big O notation.

To compute the Space complexity we use two factors constant and instance characteristics. The space requirement S(p) can be given as:

$$S(p) = C + sp$$

Where **C** is a constant i.e. fixed part and denotes the space of inputs and outputs. This space is an amount of space taken by instruction, variables and identifiers [5]. And **sp** is a space dependent upon instant characteristics. This is a variable part whose space requirement depends on particular problem instance.

1.2 Time Complexity
The time complexity of an algorithm is the amount of computer time required by an algorithm to run to completion. Time complexity required **T(P)** to run a program **P** also consists of two components:

A fixed part: compile time which is independent of the problem instance → **C**.
A variable part: run time which depends on the problem instance → **tp**(instance)

$$T(P) = C + tp(\text{instance})$$

The time complexity is measured in terms of a unit called frequency count. Frequency count is a count denoting number of times of execution of statement.
The algorithmic analysis stops here. However, the effect of using different data structures, languages, compilers, memory hierarchies, cache size and swap area etc [3]. On the performance of an implementation of an algorithm is completely ignored. From developer's point of view, all these facts contribute a lot towards the executed program. The proposed study deals with the effect of different swap areas on execution time of a program. In this study, main objective is to provide the different configurations of swap area under Linux operating system environments to improve the performance of an algorithm [8]. It also focuses on different parameter of operating system that affects execution of any type of program like swap area, cash size, page size etc. Our work is the first step to evaluate the optimize value of operating system parameter in minimum time. It also enlists the operating system parameters that affect the following components of operating system [9].

1. Memory management system
1.1 Cache size.

1.2 Swap area.
1.3 Page size.
1.4 Page fault.

2. Process management system
2.1 Turnaround time.
2.2 Time spent in user mode.
2.3 Time spent in kernel mode.
2.4 Size of process during its lifetime.
2.5 Size of process in data area.
2.6 Size of process in stack apace.
2.7 Number of times the process swapped out of main memory.
2.8 Number of context switch due to time slice expired.
2.9 Number of context switch due to I/O operations.

3. File management system
3.1 Number of file system inputs by the process.
3.1 Number of file system outputs by the process.

2. METHODOLOGY

For the customization of swap partition we develop a system that consist of two parts shell program and c program. The working diagram of system is shown below

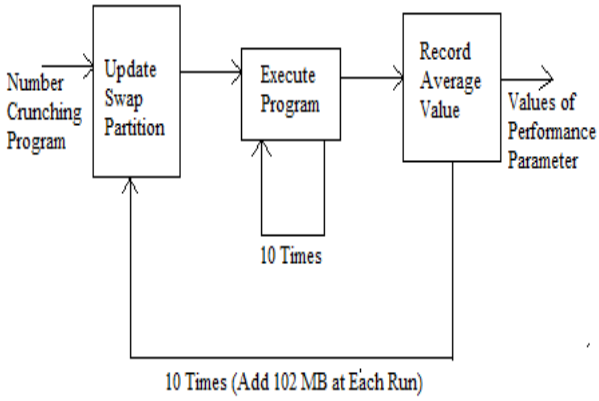

Fig 1: Working Diagram of System

The systems takes the number crunching program as input and repeats the execution 10 times for fixed value of swap partition and record the real time, MIPF, PSO, CST, CSIO. The system will repeat the same process with different value of swap partitions up to 10 times. For customization of swap area we use swap on and swap off command of Linux [10].
For the optimization of swap area at run time we develop software in C programming and some modules in shell programming language [11].

The developed software takes a C program as input, executes it with default and variable swap area and records the following values.

Real Time – Defines the time interval between submissions of process to its completion in seconds.
System Time – Defines the total time spent by a process in kernel mode.
User Time - Defines the total time spent by a process in user mode.
Waiting Time - Defines the total time spent by a process in waiting queue.
Swap Area – Defines the amount of memory required for swapping of process.
%CPU – Defines the CPU utilization during execution of process.
MIPF – Defines the number of minor, or recoverable, page faults. These are faults for pages that are not valid but which have not yet been claimed by other virtual pages.
CST- Defines the number of times the process was context-switched involuntarily (because the time slice expired).
CSIO - Defines the number of waits: times that the program was context-switched voluntarily, for instance while waiting for an I/O operation to complete.
PSO - Defines the number of times the process was swapped out of main memory.
In each run the system executes the input program ten times and computes the average. The system also increases the swap area by 102 MB in each run.

3. RESULT

We use the following number crunching function to evaluate the result

```
1. void mm(a[m][n], b[n][o], c[m][n],i, j, k)
2.       for  i=1 to m do
3.       for  j=1 to o do
4.       for  k=1 to n do
5.   C[i][j] = c[i][j]+(a[i][k]*b[k][j])
```

Where m=n=o=825

For matrix multiplication algorithm we have considered total number of elements as 825x825 that lead to total number of instruction to be executed as 825^3. These many instructions are large enough to verify the results [12]. We execute the matrix multiplication algorithm expressed in C programming language with initial value of swap area and page size as 102 MB and 4KB respectively

Table 1. Time Statistics of Matrix Multiplication for n=825

S. No.	Real Time	User Time	System Time	Waiting Time
1	21.23	18.31	0.09	2.83
2	21.25	18.21	0.08	2.96
3	21.32	18.28	0.08	2.96
4	21.07	18.16	0.09	2.82
5	21.13	18.2	0.08	2.85
6	21.06	18.15	0.1	2.81
7	21.23	18.24	0.08	2.91
8	21.25	18.1	0.07	2.08
9	21.19	18.26	0.09	2.84
10	21.12	18.19	0.07	2.86

From the table 1, we can observe that in each execution we got the slight variation in the value of real time, user time and system time [13]. This variation may occur due to parallel execution of other processes. Thus, there is need to calculate the average value of real time system time and user time. The

average value of real time, user time and system time obtained as 21.48 seconds, 18.51 seconds and 0.08 seconds respectively.

Table 2. Performance of Algorithm with Variable Swap Partition

No.	Swap Area	Real Time	%CPU	MiPF	CST	CSIO	PSO
1	102	21.31	85	2126.8	1726.5	2262.2	0
2	204	21.073	86	2125.7	1712.8	2262.5	00
3	306	21.191	87	2126.6	1717.9	2262.2	0
4	408	21.389	86.3	2126.7	1714.7	2262.1	0
5	510	21.185	86.1	2126.8	1699	2262.4	0
6	612	21.176	86	2126.8	1675.4	2262.3	447.51
7	714	21.394	86.1	2127.1	1712.6	2262.1	0
8	616	21.166	86	2126.9	1689.5	2262.2	0
9	918	21.052	86	2126.9	1683.9	2262.2	0
10	1020	21.076	86	2126.9	1699.3	2262.4	0

From the table 2, we can observe that by linear increase in swap area result in slight variation in real time, CPU utilization, MIPF, CST and CSIO. This slight variation may be occurring due to multiprogramming environment of operating system. It is also observed that for the swap area 612 MB we obtained the PSO become maximum as 447, which increase the execution time. This variation may occur due to external fragmentation in swap area.

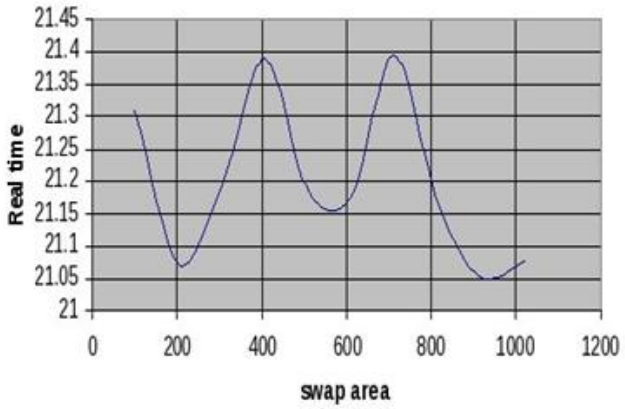

Fig 2: Swap Area Vs Real Time

Figure 2 indicates the variation in real time with respect to the swap area. This figure gives the estimation of swap area for the different value of real time. From this figure we can obtained swap area for minimum real time.

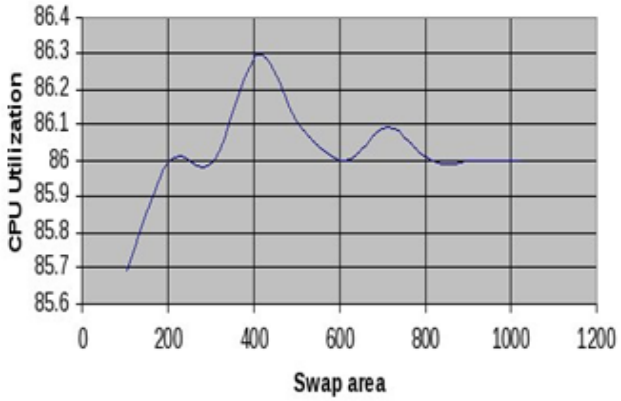

Fig 3: Swap Area Vs CPU Utilization

Figure 3 indicates the variation in CPU utilization with respect to swap area. From this figure be can obtain the required swap area and estimate for CPU utilization for the desired process execution time.

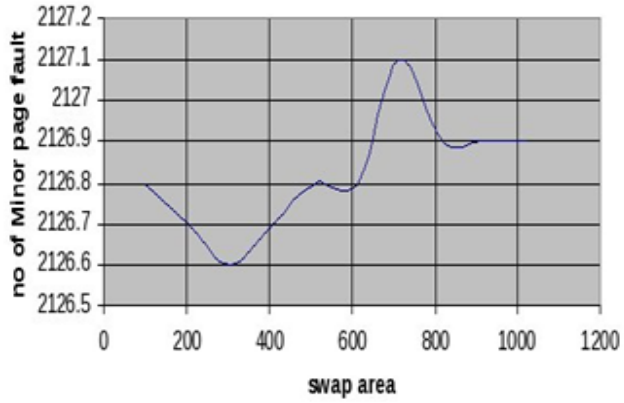

Fig 4: Swap Area Vs Minor Page Fault

Figure 4 indicates the Swap area Vs Number of minor page fault occurs during execution of algorithm. From this figure be can obtain the required swap area for minimum minor page fault for the desired process execution time.

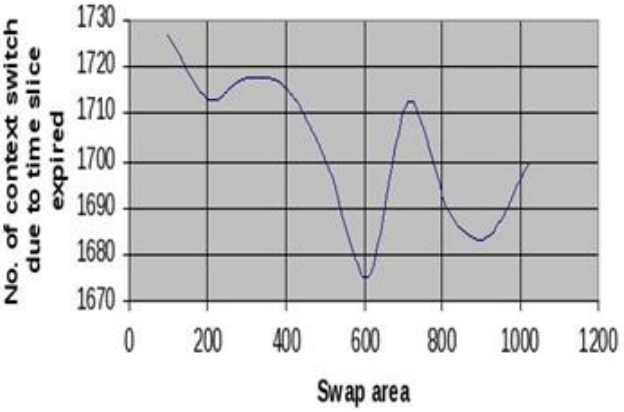

Fig 5: Swap Area Vs Context Switch due to time slice expired

Figure 5 indicates the variation in Number of context switch due to time slice expired with respect to the swap area. This figure gives the estimation of swap for the different values of context switch due to time slice expired. From this figure the swap area for minimum context switch due to time slice expired can be obtained.

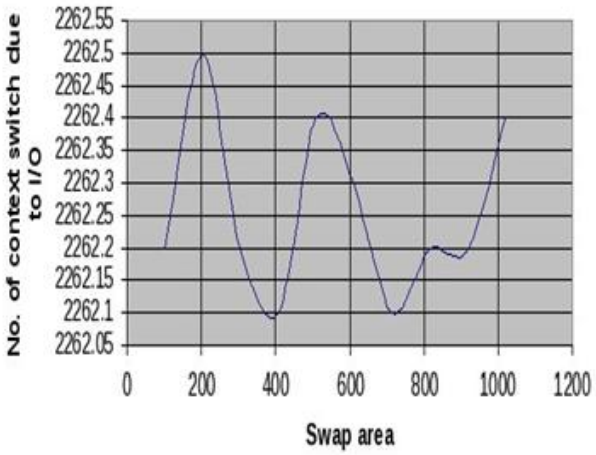

Fig 6: Swap Area Vs Context Switch due to input-output

Figure 6 indicates the variation in Number of context switch due to I/O with respect to the swap area. This figure gives the estimation of swap area for the different values of context switch due to I/O. From this figure can obtain the required swap area for Number of context switch due to I/O for the desired process execution time.

4. CONCLUSION

The analysis of algorithm is limited to their theoretical behavior of algorithm, In general we use the asymptotic notation and some rules used to estimate the execution time of language construct. Then we estimate the best case, worst case and average case performance of an algorithm for small piece of code written in a programming language. The algorithmic analysis stops here. However, the effect of using different swap area on the performance of an implementation of an algorithm is completely ignored. In this paper we use number crunching algorithm to verify the effectiveness of the developed software, we estimate the effect of swap area on the performance of given algorithm expressed in c programming language. The system records the effect of executions with respect to memory management system, Process management system, and file management system.

From the results one can obtain the following

1. Suitable swap area for maximum CPU utilization and respective execution time.
2. Suitable swap area for minimum number of recoverable page fault and respective execution time.
3. Suitable swap area for minimum number of context switch and respective execution time.

5. ACKNOWLEDGMENTS

Our thanks to my thesis guide and I thankful my parents and my little sister for their support and encouragement throughout my work. Finally I am grateful to Almighty God for his blessing for the completion of most difficult task.

6. REFERENCES

[1] S.Mansoor Sarwar, Edwin E. Parks and Syed Aqeel Sarwar "Laboratory Exercises for Practical Performance of Algorithms and Data Structures", IEEE Transactions on Education, Vol 19, No. 4, November 1996.

[2] Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajshekharan, "Performance Analysis", in Fundamentals of computer algorithms, New Delhi, India, Galgotia Publications, 2008.

[3] Sumitabha Das, "Advanced system administration", in Unix Concepts and Applications, 4 ed, New Delhi, TMH, 2006.

[4] Maurice j. Batch, "Memory Management Policies", in The Design of the Unix operating System", 6 edition, Delhi, India, Pearson Education, 2005.

[5] G.S. Baluja, "Recursion – A Breath Breaker", in Data Structures through C, 1 ed., Delhi, India, Dhanpat Rai, 2005.

[6] Udit Agrawal, "Growth of Functions", in Algorithm Design and Analysis, 1 ed., Delhi, India.

[7] Coremen Thomas, Leiserson CE, Rivest RL; Introduction to algorithms; PHI.

[8] Michael T Goodrich, Robarto Tamassia, Algorithm Design, wiely India.

[9] **S. M,** Sawar, **M,** H. **A.** Jaragh, S. **A,** Sarwar, and **J,** Brandenburg, "Engineering quicksort," Comp. Languages, to be publiced.

[10] **A.** Aho, J. Hopcroft, and J. D. Ullman, Data Structures and Algorithms. Reading, MA: Addison-Wesley, 1983.

[11] Burgin, M. Super-recursive algorithms, Monographs in computer science, Springer.

[12] ACM Curriculum Committee on Computer Science, "Curriculum '78- recommendations for the under graduate program in computer science," *Comm. ACM.*

[13] Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajshekharan, "Performance Analysis", in Fundamentals of computer algorithms, New Delhi, India, Galgotia Publications, 2008.

[14] M.L., Moore, C., and Costa, J.F. (2000) An analog characterization of the subrecursive functions. In Proc. of the 4th Conference on Real Numbers and Computers, Odense University, pp. 91–109