International Journal of Pure and Applied Mathematics
Volume 120 No. 6 2018, 4179-4192
ISSN: 1314-3395 (on-line version)
url: http://www.acadpubl.eu/hub/
Special Issue
http://www.acadpubl.eu/hub/

SCALABILITY OF MEMORY MAPPED K-MEANS ALGORITHM WITH HYPER-THREADED PROCESSORS

Dr.S.N.Tirumala Rao,
Professor & HOD,
Dept. Of Computer Science & Engineering,
Narasaraopeta Engineering College,
Narasaraopeta
nagatirumalarao@gmail.com
Sireesha Moturi,
Asst.Prof., Research Scholar
Dept. Of Computer Science & Engineering,
Narasaraopeta Engineering College, Narasaraopet
KLEF, Vaddeswaram
sireeshamoturi@gmail.com

June 21, 2018

Abstract

In the contemporary years, all CPU designers have moved from multi-core architecture to multi-core architecture with Hyper-Threading to ramp up clock speeds of multi-core processors. The conceptof memory mapped files have been widely supported by most of the contemporary operating systems. It is not un-common to employ memory mapping of files in applications involving hugeI/O bandwidth to advance the response times. This paper mainly focuses on performance evaluation of memory mapped files on i3,i5 and i7 processors usingclustering algorithms such as k-meansalgorithm with OpenMp. Experiments are carried out with serial versions and parallel versions with mmap() and fread(). Experimental results demonstrate the scalability of our implementation and effective utilization of parallel hardware, which is beneficial to Data Mining (DM) problems, which involves huge data sets. The experiments also proved that mmap() files gives improved performance compared to conventional fread() files.

1 Introduction

In the present scenario each business organization has massive competition to achieve success. So, each business organization makes strategic selections in order to attain success. Nowadays, a massive quantity of data is generating by daily activities in Science, Engg., Business and plenty of alternative areas, because of the fast advances in mechanisation and digitalisation techniques. Sometimes, users might have no proper idea on which information is more helpful. The major aim of data mining is to find hidden patterns from huge repositories. Irrespective of the application field, data mining allows to find useful hidden patterns and correlations from huge repositories which can be helpful in decision making process[1]. So, the main challenging task is to extract hidden knowledge from that voluminous data. It is essential to have a data mining system which will be able to mine various kinds of hidden patterns and knowledge to fulfill the needs of different users or applications. The data mining is a process which is able to use wide variety of tools to extract knowledge from massive datasets. The synonym of data mining is Knowledge Discovery in Databases (KDD) because it is capable of integrating different techniques form different disciplines such as neural networks, statistics, machine learning, database technology and information retrieval, etc.[1]. Data mining has several aspects like predictive modelling or classification, cluster analysis, association analysis, anomaly detection and regression analysis etc. Complexity of all these algorithms depend on time and space. With respect to the size of the dataset the complexity of these algorithms grow linearly, so these are very time consuming. There are different techniques to reduce the time requirement of CPU for the data mining applications[3,5,6]

UNIX and windows operating systems uses a special I/O technique called Memory Mapped I/O to access very large files. Memory Mapped I/O is a variant file accessing technique which is widely supported by various operating systems such as window, linux, debain, Ubuntu [7, 9, 10, 11]. In traditional file I/O, process has to switch from user mode to kernel mode to read a block of data into process address space. But, whereas in memory mapped I/O, file is directly mapped to process address space and file will be treated as an extension of virtual address space. So, memory mapped file gives better performance compared to traditional file I/O since it does not require any context switching[2,8].

In the contemporary years, all CPU designers have moved from multi-core architecture to multi-core architecture with Hyper-Threading to ramp up clock speeds of multi-core processors[4]. A multi-core processor is an integrated chip which contains two or more independent processors which run simultaneously for enhanced performance [8]. A core can be thought of as an individual processor. A dual-core processor has two internal processors fabricated into a single chip. A quad-core model has four processors i.e two dual-core processors fabricated into a single chip. More cores are useful for multi-tasking applications. For example, you can run two applications at the same time, each one having access to its own dedicated processor. More cores are also useful for multi-threaded applications such as video editing. Single-threaded applications can only use a single core leaving any others idle. Core i3 processors have two cores, i5and i7 have four cores.

Hyper-Threading is Intel's technology for creating two logical cores in each physical core. In other words, to your operating system it appears as though your CPU has double the number of cores than it really does. In terms of performance, Hyper-Threading speeds up multi-tasking and multi-threaded applications. It's not as fast or as efficient as extra 'real' cores, but it's an improvement over a single Core. A Core i3-4370 Haswell processor runs at 3.8GHz and a Core i5-4590, which only has a clock speed of 3.2GHz. A single threaded application runs faster on i3 rather than i5. However, a multithreaded applications run faster on i5 rather than i3, as its four real cores are better than the Core i3's two cores and Hyper-Threading.

In this experiment, a popular k-meansclustering algorithm is parallelized using OpenMP API(Application programming interface)[4] to exploit the capabilities of latest hardware. An experiment is carried out with k-means data mining (DM) algorithm, to explore the potential of i3, i5 and i7 architectures. The concept of memory mapped file is widely supported by most of the modern operating systems. Performance of memory mapped k-means algorithm and fread based k-means algorithm on Multi-core processor is analyzed. Experiment results with our simulated data demonstrates that the scalability of our implementation and effective utilization of parallel hardware with Hyper-Threading. Experiment results demonstrate that memory mapped algorithm gives better performance compared to the traditional file I/O.

2 Experimental setup

The parallelized k-means with fread() and mmap() have not altered the original algorithm, i.e they produced identical results as serial k-means algorithm. The k-means algorithm with fread() and with mmap() are parallelized using OpenMP API. The Parallelized K-Means algorithm with OpenMp is tested with file size of over 5GB simulated data. Computational time requirement of k-means algorithm with fread() and mmap() functions is observed with our simulated data. Computational time is measured in clock ticks. The objective of this paper is to analyse the performance improvement in mmap() based k-means clustering algorithm over fread() based k-means clustering algorithm, with respect to its execution time. Experiments are carried out with the converted binary data. The experiments are carried out on the following processors:

- Intel(R) Core^{TM}2 Quad CPU Q6600 @2.40GHz processor, 1 MB(Megabyte) Cache memory, 1 gigabyte (GB) RAM (Random Access Memory) is used in our study.

- Intel(R) Core^{TM}2 i3-4005U processor @1.70GHz, 4MB(Megabyte) Cache memory, 4 gigabyte (GB) RAM (Random Access Memory) is used in our study.
- Intel(R) Core^{TM}2 i5-24005U X 4 processor @ 1.70GHz, 4 MB(Megabyte) Cache memory, 4 gigabyte (GB) RAM (Random Access Memory) is used in our study.
- Intel(R) Core^{TM}2 i7-4510U processor 2.00GHzX4 processor, 4 MB(Megabyte) Cache memory, 4 gigabyte (GB) RAM (Random Access Memory) is used in our experiment.

Ubuntu 14.04.3, 64bit operating system is used to study the performance of parallel k-means algorithm with mmap() and without mmap().To observe the performance of parallelized k-means with fread() and mmap() we have varied number of samples, clusters and dimensions on dual core, I3, I5 and I7 processors as shown in fig 1 to 9. We could observe that mmap() is consistently taking less time than fread() independent of the dimensionality, number of samples and clusters. In order to see the performance of mmap() over fread() with varying dimensionality, clusters and number of samples,we have carried out experiments which are shown from fig 10 to 14. To see the benefit of mmap() over fread() with varying number of clusters, we have conducted experiments, which were presented from Figure 10 to 12. Experiments are carried out to see the performance of mmap() with varying number of dimensions as shown in fig 13. The observations in fig 14 indicate that the mmap() is showing its benefit over fread() independent of varying number samples. We could observe that the advantage of mmap() over fread() with all our experiments on all the selected input parameters . In all the architectures, we have observed that the mmap based k-means algorithm is more scalable with respect to number of cores and hyper threads in our implementation.

3 Conclusion

The scalability of parallelized k-means algorithm with OpenMP is studied on dual core, i3, i5 and i7 with ubuntu operating system. Experiments show that scalability of Parallelized k-means algorithm increases with number of cores. It is also observed that scalability increases with number of Hyper-Threads also. It also shows that parallelized algorithms with mmap() are more scalable than parallelized algorithms with fread() irrespective of number of samples, number of dimensions and number of clusters. Our observations also reveal that the computational benefit of mmap() over fread() based algorithms is independent of number of dimensions, number of samples and number of clusters. The advantage of mmap() increases as number of cores and number of Hyper-Threads increases.

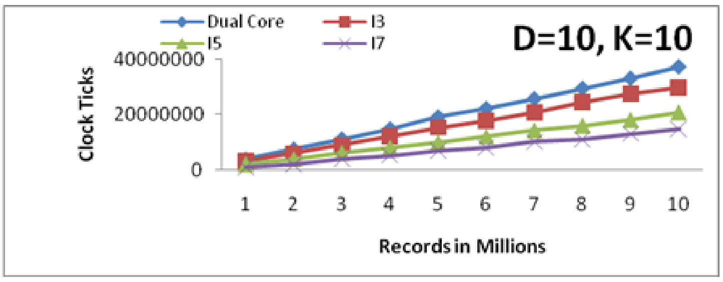


Fig1.k-means Algorithm with fread() for Random Data set of 10 Dimensions and 10 Clusters.

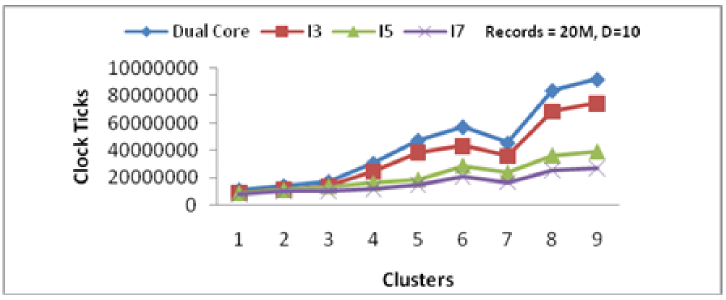


Fig2. k-means Algorithm with fread() for Random Data Set Data of 20 Million Records and 10 Dimensions

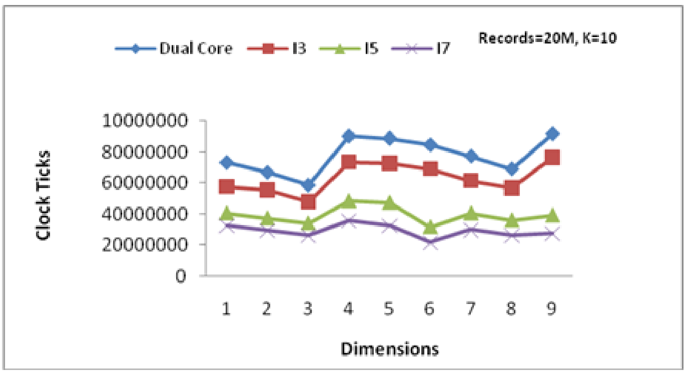
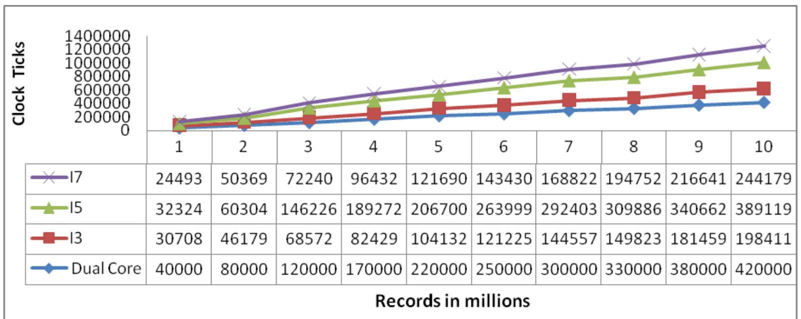


Fig3. k-means Algorithm with fread() for Data Set Data of 20 Million Records and 10 Clusters.



	1	2	3	4	5	6	7	8	9	10
I7	24493	50369	72240	96432	121690	143430	168822	194752	216641	244179
I5	32324	60304	146226	189272	206700	263999	292403	309886	340662	389119
I3	30708	46179	68572	82429	104132	121225	144557	149823	181459	198411
Dual Core	40000	80000	120000	170000	220000	250000	300000	330000	380000	420000

Fig4. Time required for fread() in k- means algorithm for Random data set data of 10 Dimentsions and 10 Clusters.

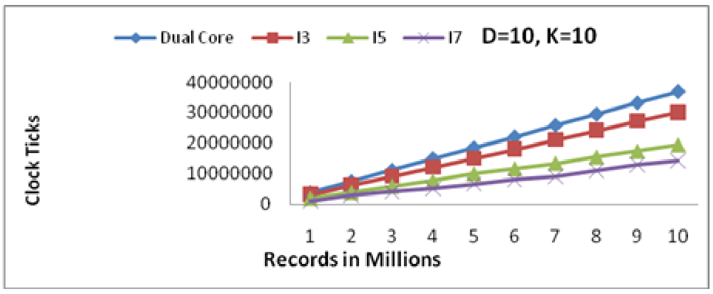


Fig 5. k-means Algorithm with mmap() for Random Data set of 10 Dimensions and 10 Clusters.

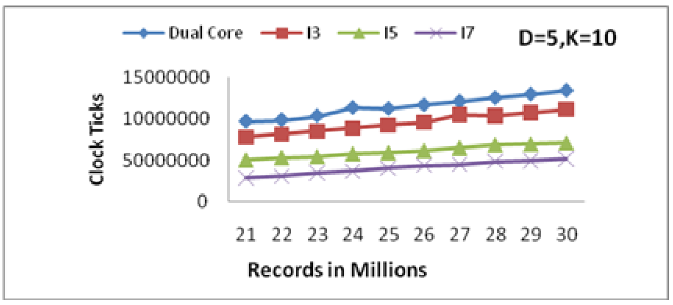


Fig 6. Total time taken for k-means Algorithm with mmap() for Random Data set of 5 Dimensions and 10 Clusters.

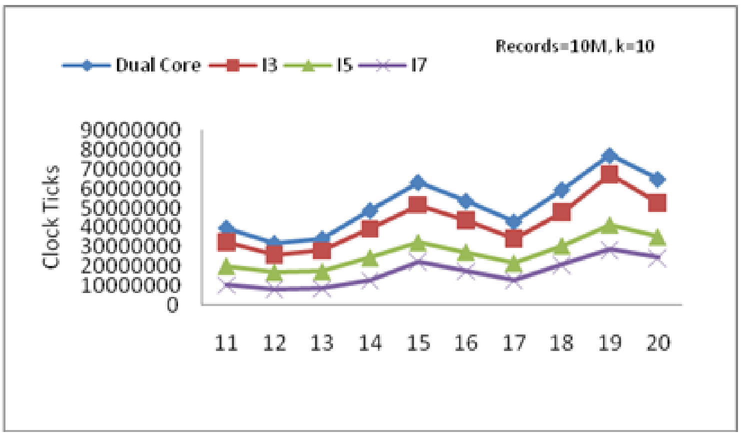


Fig 7. Total time taken for k-means Algorithm with mmap() for Random Data set of 10 million records and 10 Clusters.

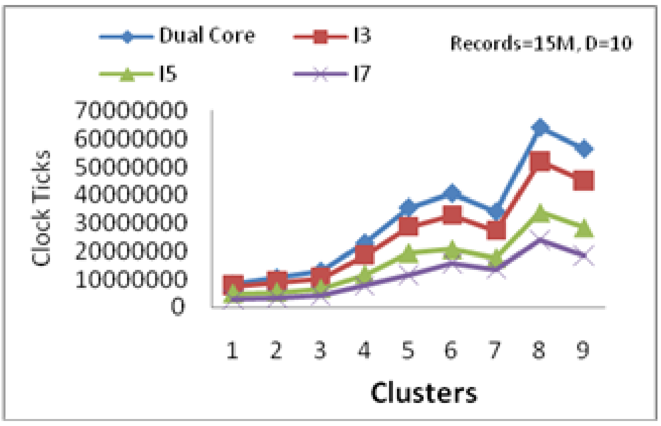
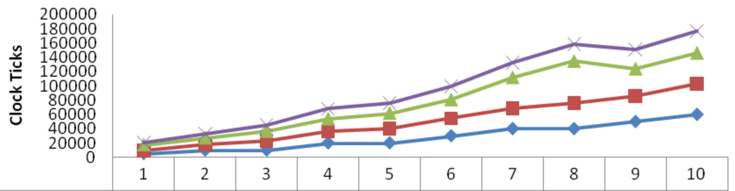


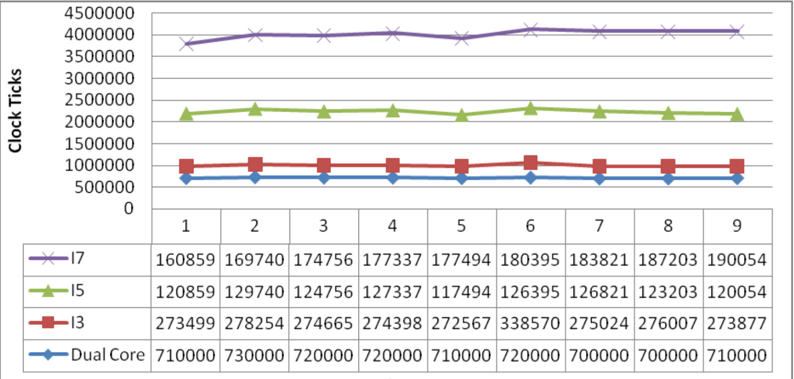
Fig 8. Total time taken for k-means Algorithm with mmap() for Random Data set of 15 million records and 10 dimensions.



	1	2	3	4	5	6	7	8	9	10
I7	3342	5856	9069	14612	14689	18623	20550	23284	27028	30697
I5	7886	8446	13377	17329	21393	26234	43397	58759	38341	43046
I3	4438	8568	13030	16418	20315	24551	28426	35977	35665	43234

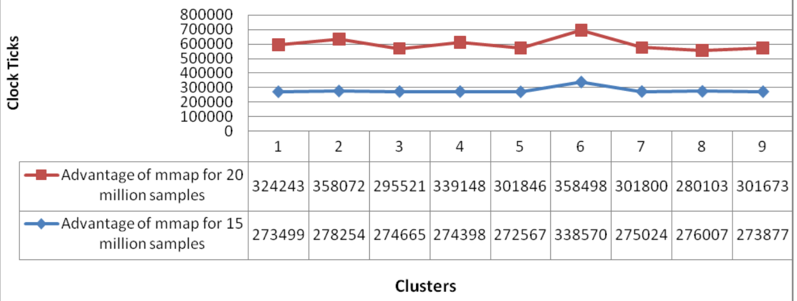
Records in millions

Fig9. Time required for mmap() in k- means algorithm for Random data set data of 10 Dimentsions and 10 Clusters.



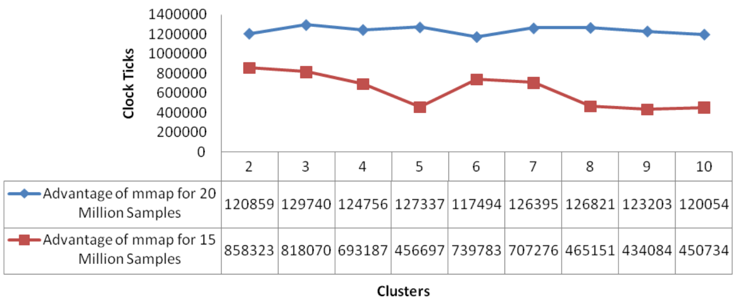
	1	2	3	4	5	6	7	8	9
✕ I7	160859	169740	174756	177337	177494	180395	183821	187203	190054
▲ I5	120859	129740	124756	127337	117494	126395	126821	123203	120054
■ I3	273499	278254	274665	274398	272567	338570	275024	276007	273877
◆ Dual Core	710000	730000	720000	720000	710000	720000	700000	700000	710000

Fig 10. Advantage of mmap() over fread for 20 Million samples with Dimentionality 10 in Random data set in K-means algorithm.



	1	2	3	4	5	6	7	8	9
■ Advantage of mmap for 20 million samples	324243	358072	295521	339148	301846	358498	301800	280103	301673
◆ Advantage of mmap for 15 million samples	273499	278254	274665	274398	272567	338570	275024	276007	273877

Fig11. Advantage of mmap over fread for 15 & 20 Million Samples with Dimention 10 in Random data set in K-means algorithm on Dual Core Processor



	2	3	4	5	6	7	8	9	10
◆ Advantage of mmap for 20 Million Samples	120859	129740	124756	127337	117494	126395	126821	123203	120054
■ Advantage of mmap for 15 Million Samples	858323	818070	693187	456697	739783	707276	465151	434084	450734

Fig12. Advantage of mmap over fread for 15 & 20 Million Samples with Dimention 10 in Random data set in K-means algorithm on I5 Processor

10

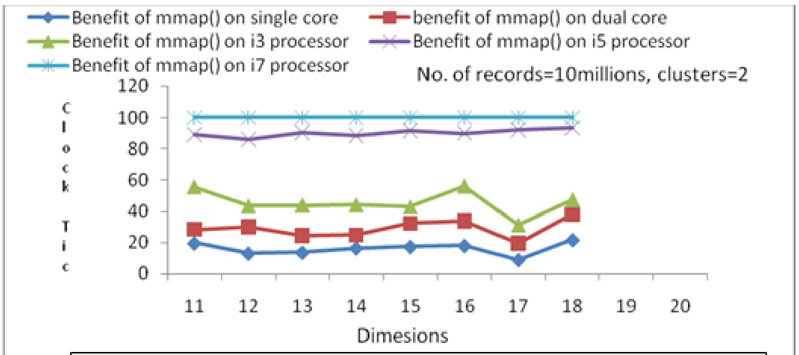


Fig13. Advantage of mmap()over fread() for 10 Million Samples with 2 clusters in Random data set with K-means algorithm on single core, dual core, I3, I5 and I7 processors

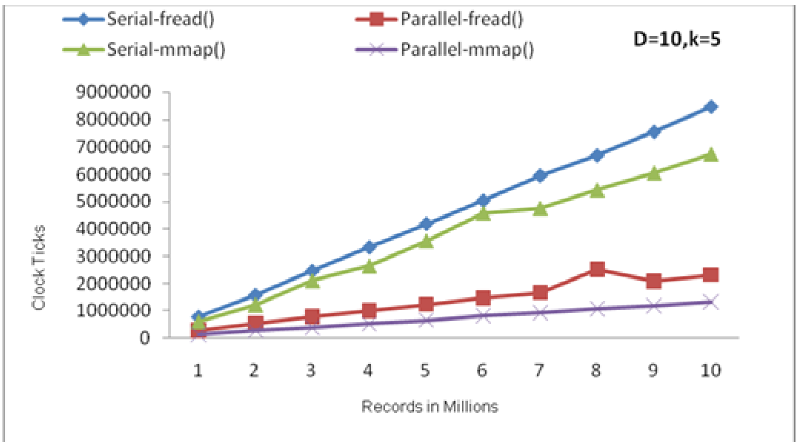


Fig14. Advantage of parallel fread() over serial fread() and parallel mmap() over serial mmap() for 10 dimensions and 5 clusters in Random data set with K-means algorithm

References

[1] Han.J, Kamber.M, “Data Mining: Concepts and Techniques”, Morgan kaufmann Publishers, Book, 2000.

[2] S .N. Tirumala Rao, E. V. Prasad, N. B. Venkateswarlu and B. G. Reddy, “Significant performance evaluation of memory mapped files with clustering algorithms”, IADIS International conference on applied computing, Portugal pages .455-460, 2008.

[3] A.Gray and A.More, july 2004, “Data Structures for fast statistics International conference On Machine learning”, Alberta Canada.

[4] S .N. TirumalaRao, E. V. Prasad, N. B. Venkateswarlu, “A Scalable k-means Clustering Algorithm on Multi-Core Architecture”, IEEE International Conference On Methods And Models In Computer Science,(ICM2CS-09), JNU Delhi, December 14-15 2009, India.

[5] ECE 222 System Programming Concepts lecturer notes on system calls, www.parl.clemson.edu.

[6] N.B.Venkateswarlu, M.B.Al-Daoud and S.A Raberts, “Fast K-Means Clustering Algorithms”, University of Leads School of Computer Studies Research Report Series Report 95.18.

[7] Brain W.Keringhan, Dennis M.Ritchie,2nd edition, 2004, “C Programming Language”, PHI publications, New Delhi 110 001, India.

[8] https://searchdatacenter.techtarget.com/definition/multi-core-processor.

[9] Laurent, Anne,et al. pgp-mc: Towards a multi-core parallel approach for mining gradual petterns . Database systems for advanced applications. Springer Berlin/Heidelberg, 2010.

[10] Uresh vahalia, 2008, UNIX Internals The New Frontiers, Pearson education, New Delhi 110 017, India.

[11] Maurice J.Bach ,2004, The design of Unix operating system,
PHI publications, New Delhi 110 001, India.

[12] https://www.intel.com/content/www/us/en/products/processors/core.html.