Volume 3, Issue 6, June 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Parallelizing Apriori on Hyper-Threaded Multi-Core Processor

Anuradha.T *
Department of ECM,
KL University, India

Satya Prasad.R
Department of CSE,
Acharya Nagarjuna University, India

Abstract*— With the ever growing demand for increasing the processing power, CPU manufacturers have introduced different processor designs . Hyper- Threading, which is Intel's trade mark for simultaneous multithreading technology increases the processing power of a single core processor by allowing it to run two threads by adding a virtual thread along the side of a physical thread . As it is difficult to further increase the clock frequencies of a single core processor after a certain limit, multi-core processors have emerged. They can increase the processing performance by packaging more than one physical core on a single chip. The new generation Intel Core iX series processors combined the concept of hyper-threading(HT) with multi-core technology to further increase the performance of multi-core processors. But, however powerful the hardware may be, software applications can obtain the potential performance improvements from these hardware advancements only if they are properly threaded by parallelization techniques. The experimentation was conducted to evaluate the performance of apriori, a popular data mining algorithm on a hyper threaded dual-core processor compared to the performance on a non hyperthreaded dual-core processor with two different Linux functions for accessing the data from a file, namely fread() and mmap(). The experiments proved that the speedup and efficiency of our parallel algorithm were more on a hyperthreaded processor compared to the speedup and efficiency on a non hyper threaded processor with both fread() as well as mmap().

***Keywords*— apriori, data mining, fread(), Hyper-Threading, Intel DualCore processor, Intel Corei3 processor, mmap(), Multi-core, OpenMP**

I. INTRODUCTION

Before the introduction of hyper-threading, only one instruction stream at a time could be executed by a single CPU by maintaining a single instruction pointer. And the processor is allowed to execute multiple threads by switching between the threads. Later the simultaneous multithreading technology (SMT) was developed where a single processor could execute multiple threads simultaneously without switching.[1] Intel Pentium 4 architectures adopted SMT with their trade mark called Hyper-Threading technology(HT technology).[2].A single processor will appear as multiple logical processors with HT technology. So operating systems can schedule multiple threads to these logical processors in the way how threads can be scheduled on multiprocessor systems. Although HT technology maximizes the performance of a single processor by effective utilization of the processor resources, there is a limiting factor as the logical processors share most of the basic resources like cache, FPU and integer math unit of the single physical processor.[3] And as the computer industry has also identified that the future performance improvements could not come from making a single core faster by making it complex, it must come largely by increasing the number of cores on a single chip, the later development in the processor architecture called the multi-core architecture has evolved.[4],[5]. By integrating more than one processor on a single chip, multi-core processors improve the performance of parallel applications with less communication cost among the cores. The new generation Intel 'core iX ' series processors combined the architectural features of both multi-core and Hyper-threading technologies to make each core on the die more faster [6]. Though these architectures are expected to bring significant execution speedups for the applications [7], not all the applications may get the performance benefits from these architectures as with any hardware feature[8],[9]. If the application need to get the benefit from these new processors, it should be a well written multithreaded application[3]. Our previous research was focused on analysing the performance of apriori by parallelizing it with OpenMP threads on a non hyper threaded Intel Pentium DualCore processor [10],[11] by using the conventional Linux fread() function to read the data from the data file. Then the performance of parallel apriori was also tested on Pentium DualCore by using the concept of memory mapping of files to get the data from the file with Linux mmap() function. The performance of serial and parallel apriori with mmap() compared to fread() was done in [12]. To identify the performance of parallel apriori on a hyper threaded processor, the same experimentation with both fread() and mmap() was done on Intel Corei3 processor and the results were compare with the results obtained in our previous research[10],[11],[12].

II. HYPER-THREADING ON A SINGLE-CORE

Hyper-Threading allows a single core processor to execute two threads by adding a "virtual" thread alongside the physical thread. From the architectural point of view, a hyperthreaded processor contains two logical processors. Each logical processor has its own processor architectural state which consists of all the processor registers like data, Segment, control ,debug and other model specific registers to avoid the context switch between the two threads [13]. Each processor also has independent instruction streams , own APIC(advanced programmable interrupt controller) and they share the execution resources consisting of execution engine, the caches, the system bus and the firmware of the processor core [14],[15],[16]. Each logical processor handles the interrupts sent to it on its own as it has its own APIC. By presenting itself to the operating system as two identical virtual processors, a hyper-threading processor will handle the work load generated by a CPU-intensive operation effectively because of the two logical processors handling the tasks at the same time [8].

III. Hyper-threading on Dual-Core

Being its own set of execution units, a core is different from a logical processor and in hyperthreaded Dual-Core, each core constitutes hyper-threading technology, enabling two threads to run simultaneously on each core with a little hardware overhead . So, in a dual-core HT Technology enabled system, one physical processor chip can have two cores and four logical processors. The architectural register state is duplicated on all the four logical processors and all of them share the physical execution resources[2],[17]. Figure 1 shows the block diagram of a hyperthreaded dual-core processor (Fig 1) [16],[18]. From the software point of view all the cores are functionally equivalent and four separate threads can simultaneously run on the four logical cores which is good for multi-threading applications[15]. From a micro architecture perspective, this means that instructions from all the logical processors will persist and execute simultaneously on shared execution resources [2]. In a computer system with hyperthreaded dual-core processor , the performance monitor shows the usage of all the four logical processors. Fig 2 and Fig 3 shows the performance monitors of non hyperthreaded dual-core(Intel Pentium DualCore)and a hyperthreaded dual-core processor (Intel Corei3) systems. We can see the CPU usage history showing two and four processors respectively on these two systems(Fig 2, Fig 3).

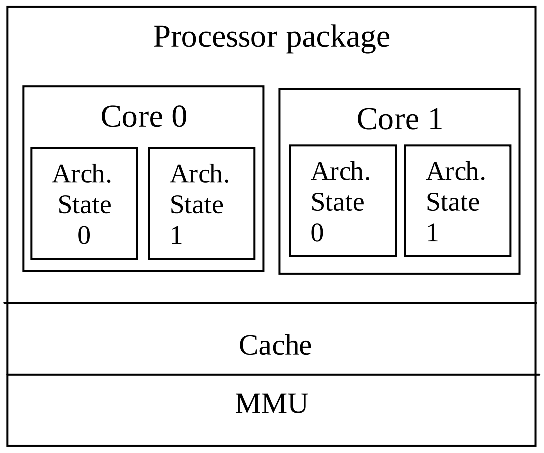
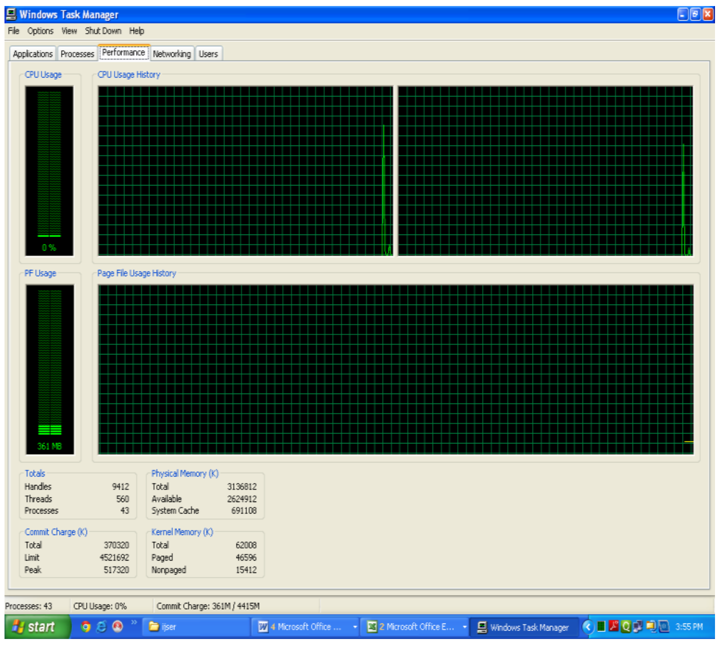


Fig 1: Dual core processor with Hyper-Threading



Totals	
Handles	9412
Threads	560
Processes	43

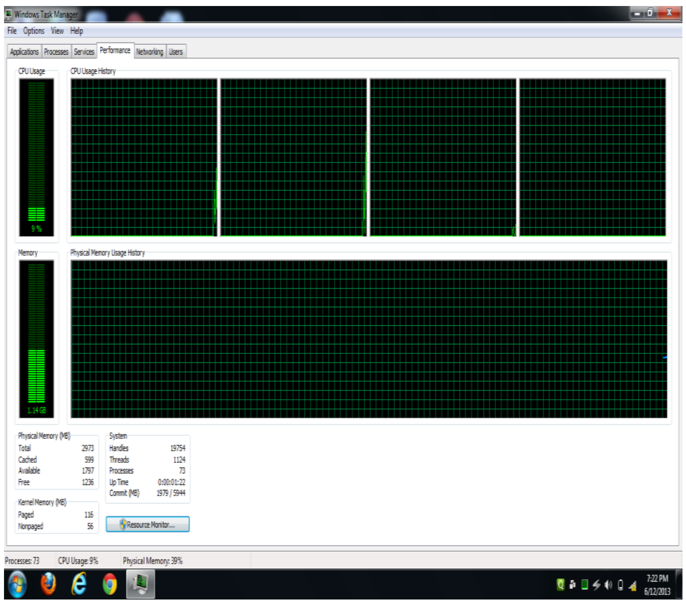
Physical Memory (K)	
Total	3136812
Available	2624912
System Cache	691108

Commit Charge (K)	
Total	370320
Limit	4521692
Peak	517320

Kernel Memory (K)	
Total	62008
Paged	46596
Nonpaged	15412

Processes: 43 CPU Usage: 0% Commit Charge: 361M / 4415M

Fig 2: Screenshot of CPU performance monitor of non hyperthreaded Pentium DualCore processor showing two CPUs in CPU usage history



Physical Memory (MB)
Total 2973
Cached 599
Available 1797
Free 1236

Kernel Memory (MB)
Paged 116
Nonpaged 56

System
Handles 19754
Threads 1124
Processes 73
Up Time 0:00:01:22
Commit (MB) 1979 / 5944

Processes: 73 CPU Usage: 9% Physical Memory: 39%

Fig 3: Screenshot of CPU performance monitor of hyperthreaded Corei3 processor showing four CPUs in CPU usage history

IV. RELATED WORK.

There was only a little research done on identifying the effect of HT technology on the performance of applications. HT performance evaluation using OpenMP threaded matrix matrix multiplication and a bench mark of 256-particle molecular dynamics on a dual processor –server with HT was conducted by Boisseau et al [19]). Characterization of Java application performance using Pentium 4 processors with HT was done by Haung et al. [20]. Utilization of Pentium 4 performance counters in the performance of garbage collection in HT mode was studied by Blackburn et al[21]. Saini et al., [22] studied the impact of HT on processor resource utilization . They have also investigated how the scientific application performance in HT mode was affected because of shared resources and the utilization of these resources was compared between single threaded(ST) and HT modes. They have conducted the experiments on four NASA applications and investigated that HT technology improved the processor resource utilization more but this may not result in overall application performance gain. Gasmi et al.,[23] proposed MTFASTFDS, a multi threaded parallel algorithm on a hyper-threading multi core processor for mining functional dependencies. They have shown that their parallel algorithm scales very well with the number of cores available. Tian et al., [24] conducted the performance study on Hyper-Threading technology (HT) enabled Intel single processor and multi-processor systems by parallelizing two multimedia applications with OpenMP pragmas. They have presented the threaded code generation and optimization techniques and concluded that multithreaded code with OpenMP pragmas yields more speedup on a HT-enabled dual-CPU compared to HT-enabled single-CPU.

V. EXPERIMENTAL WORK

The present experimental study mainly concentrates on identifying the performance of a popular data mining algorithm apriori on a hyperthreaded processor. Apriori is a frequent itemset mining algorithm which finds the frequent itemsets in the transactional data base[25].Frequent itemsets are the combination of items in the given transactional database which occur more than a pre specified number (minimum support count) of times in the database. Our work concentrates on parallelizing the algorithm using OpenMP threads and analysing it by changing the number of threads from 2-4 [10],[11].The performance of the algorithm was also analysed by using two different I/O mechanisms –one with Linux fread() and the other with Linux mmap() function. The fread() is a traditional and high level I/O function to read the data from a file where mmap() is a special and low level I/O function which gets the data from the file by directly mapping it to the process address space[26],[12]. Intel CORE^{TM}i3 processor which is a Hyper-Threading technology (HT) enabled dual-core processor with speed 2.10GHz and 3GB RAM, was used in the experimentation. Fedora 17 Linux(kernel 3.5.2-3.fc17.X86_64,Red Hat nash version 4.5.0-5) equipped with GNU C++(gcc version 4.7) was used to get the OpenMP compatibility. The parallelization was done using data parallel strategy by partitioning the database into number of partitions equal to number of threads. OpenMP fork–join model and work sharing constructs were used for parallelization [27],[28],[10],[11]. Five different synthetic data sets with 2, 4, 6, 8, 10 lakh records which resemble the transactional data base of a retail store and the standard accident dataset from the UCI data repository[29] were used. The synthetic datasets consists of randomly generated records and each record consists of a minimum of 1 and maximum of 10 items and may have any combination of a sub set of items from item1 to item10 and in the accident dataset also we have used only the first 10 items. Time taken for the execution of the serial and parallel implementations was obtained from the real time parameter of Linux time command. The results obtained on the Intel CORE^{TM}i3 processor were compared with the results obtained on the Intel Pentium DualCore processor [10],[11],[12] which was a non hyper threaded processor.

VI. Experimental Results

A. Notations Used

The following table represents the notations used in presenting the experimental results.

TABLE I
NOTATIONS USED

Notation used	**Meaning**
nrl	Number of records in lakhs
pmsc	Percentage of minimum support count
SAFD	Serial apriori with fread() on Pentium DualCore
PAFD	Parallel apriori with fread() on Pentium DualCore
SAMD	Serial apriori with mmap() on Pentium DualCore
PAMD	Parallel apriori with mmap() on Pentium DualCore
SAFi3	Serial apriori with fread() on Corei3
PAFi3	Parallel apriori with fread() on Corei3
SAMi3	Serial apriori with mmap() on Corei3
PAMi3	Parallel apriori with mmap() on Corei3
prmbD	Percentage of real time mmap() benefit on Pentium DualCore
Prmbi3	Percentage of real time mmap() benefit on Corei3
2TH,3TH,4TH	2 Threads, 3 Threads, 4 Threads

B. Experimental Results by keeping nrl fixed

The following observations were made from our experimental results by changing the minimum support count and keeping the dataset fixed:

- As the support count increases, execution time for both serial and parallel apriori with fread() as well as mmap() functions decreased on corei3. (Fig 4, Fig 5). Similar observation was found on Pentium dual-core processor [10],[11],[12]
- Execution time of parallel apriori was less compared to the execution time of serial apriori in the case of both fread() and mmap()on core i3 (Fig 4,Fig 5).Similar observation was also made with Pentium DualCore. ([10],[11],[12]).
- Real time values of the fig.s 4 and 5 (Y-axis values) indicate that the execution times of serial and parallel apriori with mmap() were less than the execution times of serial and parallel apriori with fread() on Corei3. This was also true in the case of Pentium DualCore[12].

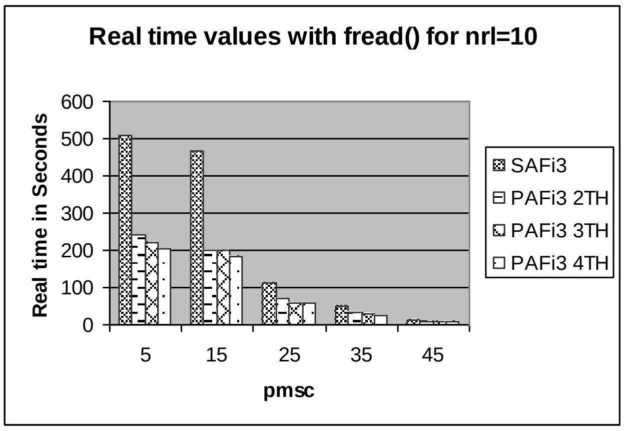


Fig 4:Real time values of serial and parallel apriori with fread() on core i3 for 10 lakh dataset

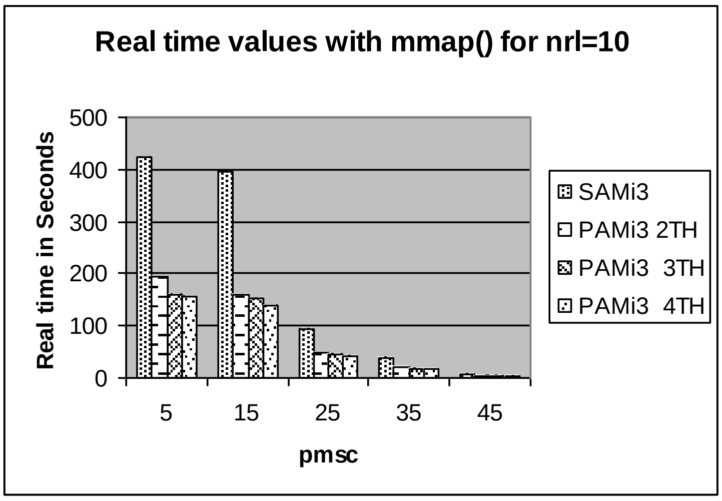


Fig 5: Real time values of serial and parallel apriori with mmap() on core i3 with 10lakh dataset

1) Comparison of real time results Corei3 Vs Pentium DualCore:

- The real time values of serial and parallel apriori with fread() on Corei3 were compared to the serial and parallel real time values with fread() on Pentium DualCore. The results indicated that the execution times for both serial and parallel apriori were less on Corei3 at all the support counts. (Fig.6,Fig.7) From fig 7, we can also derive that on Pentium DualCore, as the number of threads increases real time increased whereas on Corei3, real time decreased with the increase in the number of threads at all the support counts. This can be attributed to the fact that in the case of Pentium DualCore when the number of threads are more than 2, they need to share the existing processors and there will be a context switch overhead. Whereas on Corei3, 4 threads can run simultaneously on 4 logical processors and there was no context switch overhead upto 4 threads.
- The above mentioned trend was also observed with mmap() at all the support counts when we compare the results of serial and parallel apriori with mmap() on Corei3 with the results of serial and parallel apriori with mmap( ) on Pentium DualCore.(Fig 8,Fig 9)

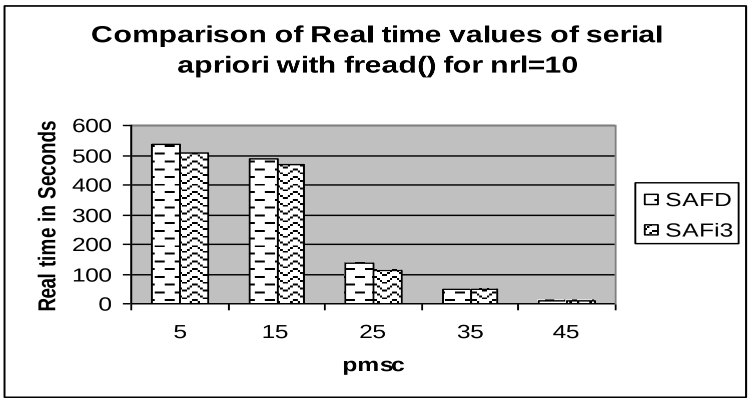


Fig 6: Comparison of real time values of serial apriori with fread() on Pentium DualCore Vs Corei3 for 10lakh dataset

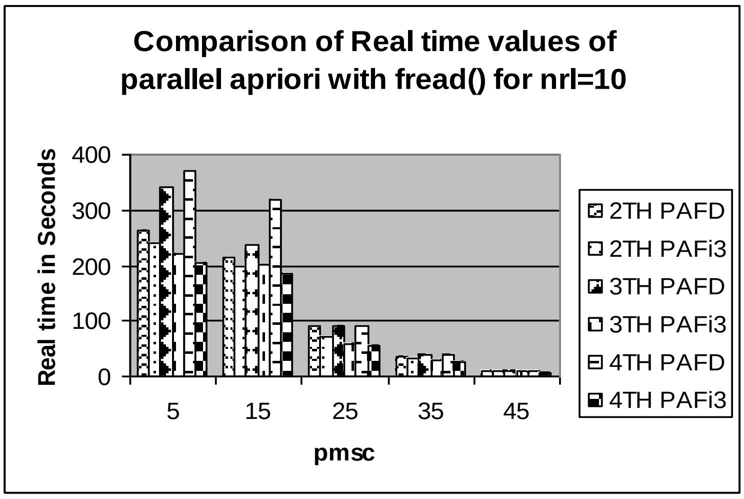


Fig 7: Comparison of real time values of parallel apriori with fread() on Pentium DualCore Vs Corei3 for 10lakh dataset

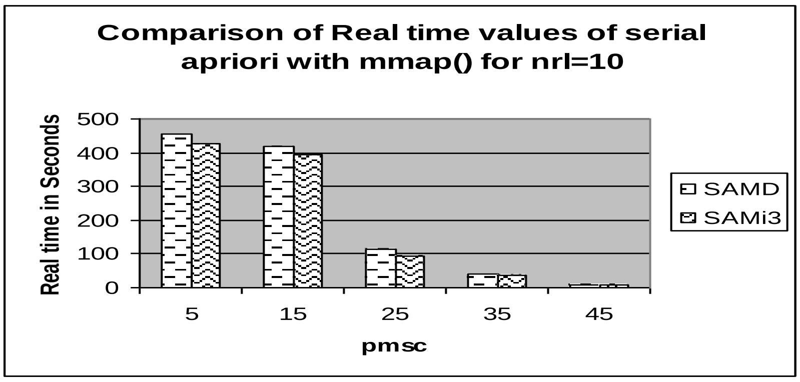


Fig 8: Comparison of real time values of serial apriori with mmap() on Pentium DualCore Vs Corei3 for 10lakh dataset

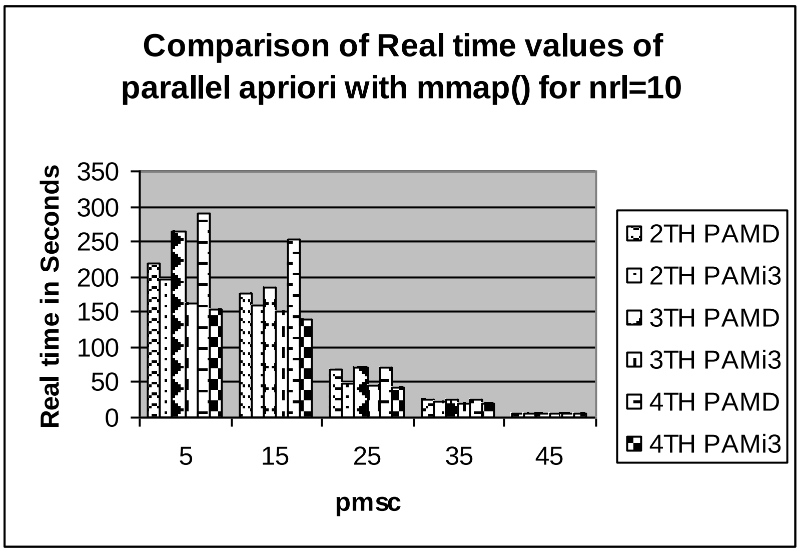


Fig 9: Comparison of real time values of parallel apriori with mmap() on Pentium DualCore Vs Corei3 for 10lakh dataset

2) Comparison of the performance of parallel apriori on Corei3 Vs Pentium DualCore

The performance of the parallel apriori was measured based on speedup and efficiency by using the following formulas where P is the number of physical processors[30].

$$Speedup_P = \frac{Execution\,time\ of\ sequential\ a\lg orithm}{Execution\,time\ of\ parallel\ a\lg orithm}$$

$$Efficiency_P = \frac{Execution\ time\ of\ the\ sequential\ a\lg orithm}{P \times Execution\ time\ of\ the\ parallel\ a\lg orithm}$$

- Speedup of parallel apriori with fread() on Corei3 was more compared to the speedup on Pentium DualCore at all the support counts. And in the case of Pentium DualCore, the speed up of parallel implementation with fread() decreased as the number of threads increased whereas in the case of Corei3 , speed up increased with the increase in the number of threads (fig 10) .This may be because on Corei3 parallel real time values decreased with the increase in the number of threads and they were increased with the increase in the number of threads on Pentium DualCore.
- To compare the speedup obtained by using mmap() over fread(), speedup of parallel apriori with mmap() was calculated by using the real time values of parallel apriori with mmap() and the corresponding serial real time values with fread().Speedup of parallel apriori with mmap() on Corei3 was more compared to the speedup on Pentium DualCore at all the support counts. And in the case of Pentium DualCore, the speed up of parallel implementation

with fread() decreased as the number of threads increased whereas in the case of Corei3 , speed up increased with the increase in the number of threads (fig 11).This may be attributed to the trend of parallel real time values explained above.

- From figs.10 and 11 , we can observe that the speedup of parallel apriori obtained with mmap() was found to be more when compared to the speedup obtained with fread() at all the support counts on both the processors.
- Efficiency of parallel apriori on Corei3 was found to be more compared to the efficiency on Pentium DualCore for both fread() and mmap() at all the minimum support counts. And the efficiency of parallel apriori decreased when the number of threads increased on Pentium DualCore but the efficiency of parallel apriori increased with the increase in the number of threads on Corei3 for both fread() and mmap().(Fig 12,13) And from the figures we can also observe that the efficiency with mmap() was more compared to the efficiency with fread() on both the processors at all the minimum support counts.

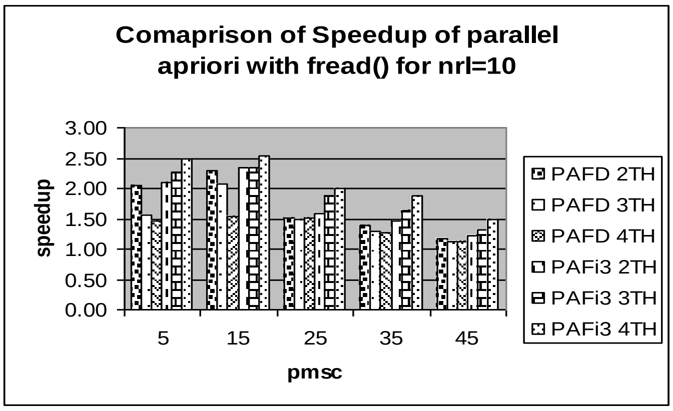


Fig 10: Comparison of speedup of parallel apriori with fread() for 10 lakh dataset on Pentium DualCore Vs Corei3

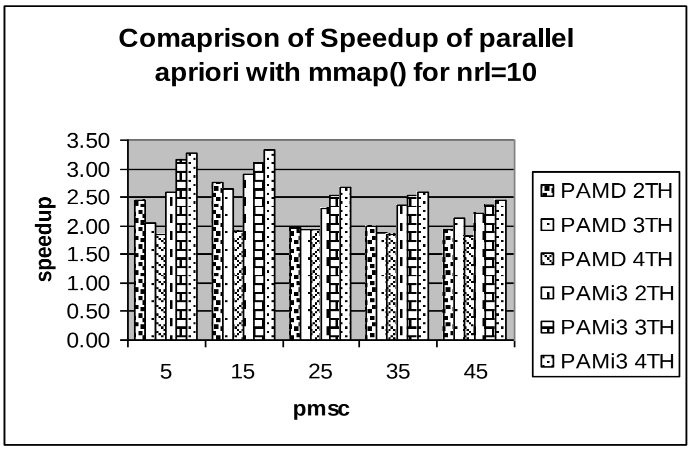


Fig 11: Comparison of speedup of parallel apriori with mmap() for 10 lakh dataset on Pentium DualCore Vs Corei3

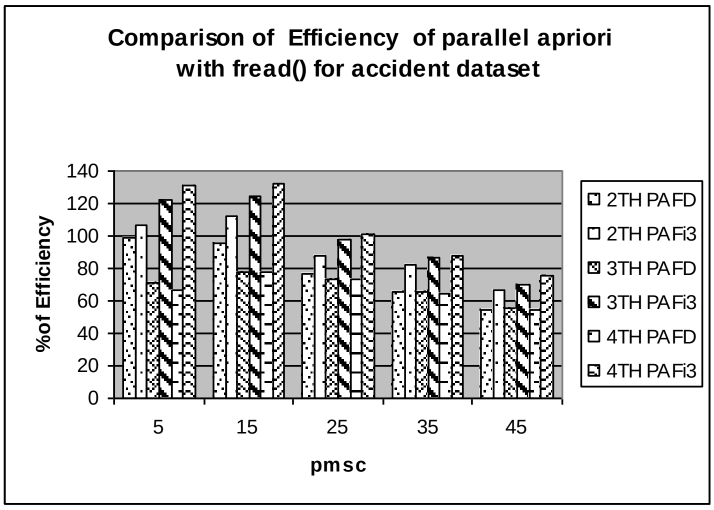


Fig 12: Comparison of efficiency of parallel apriori with fread() for accident dataset on Pentium DualCore Vs Corei3

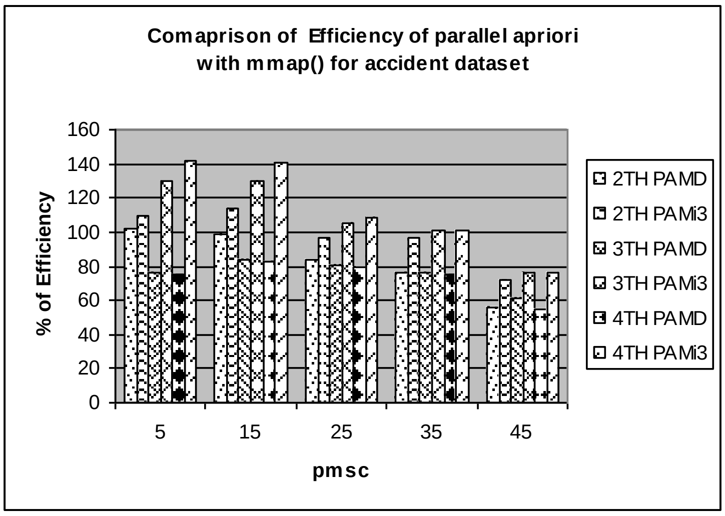


Fig 13: Comparison of efficiency of parallel apriori with mmap() for accident dataset on Pentium DualCore Vs Corei3

3) Comparison of the real time mmap() benefit on Corei3 Vs Pentium DualCore

The benefit obtained by using mmap() over fread() was computed using the following formula

$$\% \, of \ real\,time \ mmap() \ benefit = \frac{Execution\,time \ with \ fread() - Execution\,time \ with \ mmap()}{Execution\,time \ with \ fread()} \times 100$$

- o Percentage of real time mmap() benefit (prmb) was more for parallel apriori compared to serial apriori on both Pentium DualCore and Corei3 processors and the mmap() benefit increased with the increase in the minimum support count for both the processors.(Fig 14)
- o Percentage of real time mmap() benefit values on core i3(prmbi3) were more compared to prmb values on Pentium DualCore (prmbD) at all the support counts.(Fig 14)

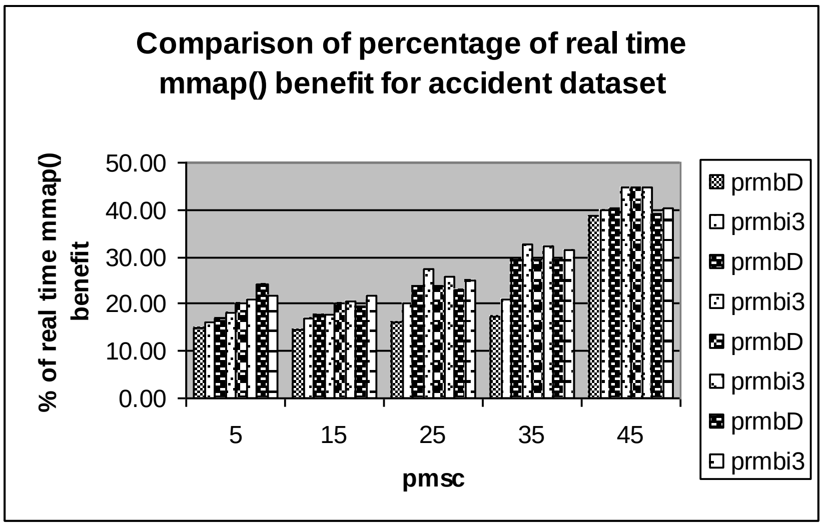


Fig 14: Comparison of real time mmap() benefit of parallel apriori with mmap() on Pentium DualCore Vs Corei3 for accident dataset

C. *Experimental Results by keeping pmsc fixed*

The following observations were made from our experimental results by changing datasets and keeping the minimum support count constant.

- o As the dataset size increases, execution time increased for both serial and parallel apriori with fread() as well as mmap()on both Pentium DualCore and Corei3. And the real time values with fread() and mmap() on hyperthreaded Corei3 were less than the real time values on non hyperthreaded Pentium DualCore .(Fig.s 15,16)
- o For all the datasets, parallel real time values were less than the serial real time values for both fread() and mmap(). And the real time values of parallel apriori increased with the increase in the number of threads for Pentium

DualCore and the parallel real time values decreased with the increase in the number of threads for Corei3. (Figs 15,16)

- Efficiency and speedup of parallel apriori were found to be more or less same for different datasets at a particular support count on both Pentium DualCore and Corei3 with fread() as well as mmap(). (Fig.s 17,18)
- The efficiency and speedup of parallel apriori with mmap() was more compared to the efficiency and speedup of parallel apriori with fread() for all the datasets.

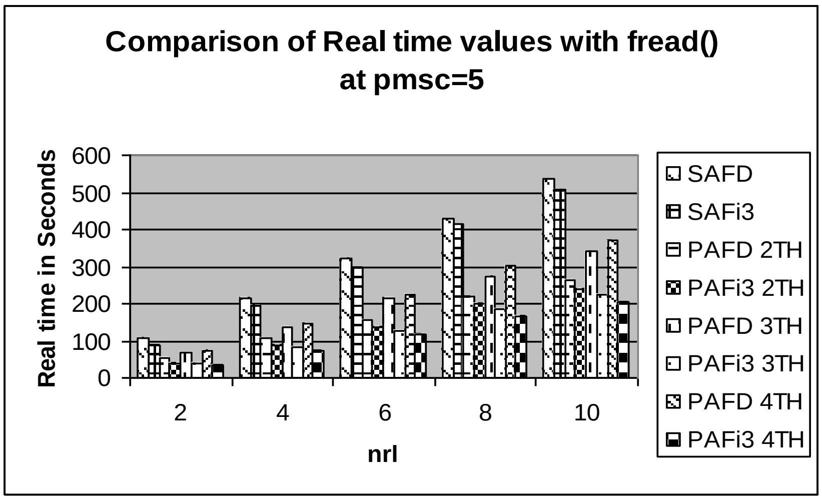


Fig 15: Comparison of real time values of serial and parallel apriori with fread() on Pentium DualCore Vs Corei3 at pmsc=5

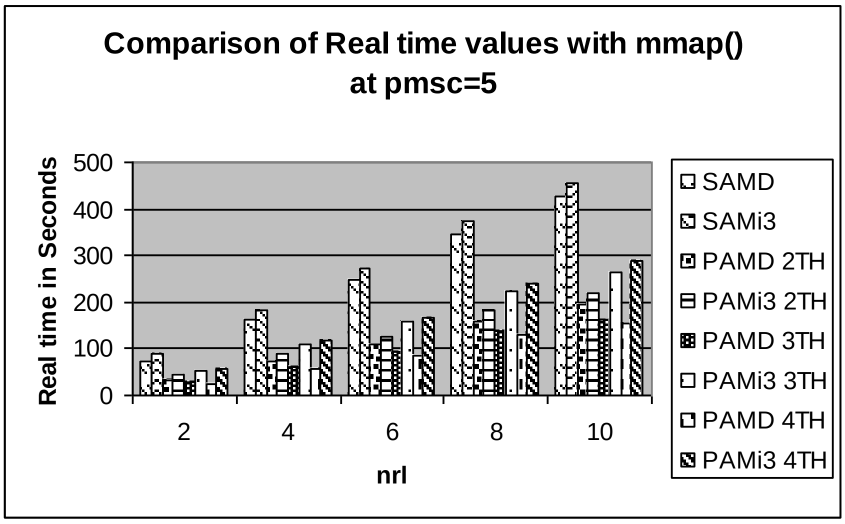


Fig 16: Comparison of real time values of serial and parallel apriori with mmap() on Pentium DualCore Vs Corei3 at pmsc=5

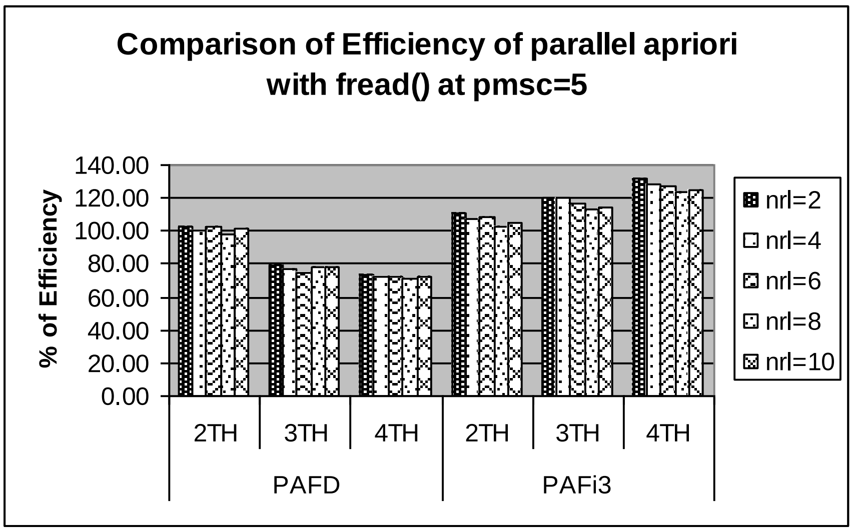


Fig 17: Comparison of Efficiency values of parallel apriori with fread() on Pentium DualCore Vs Corei3 at pmsc=5

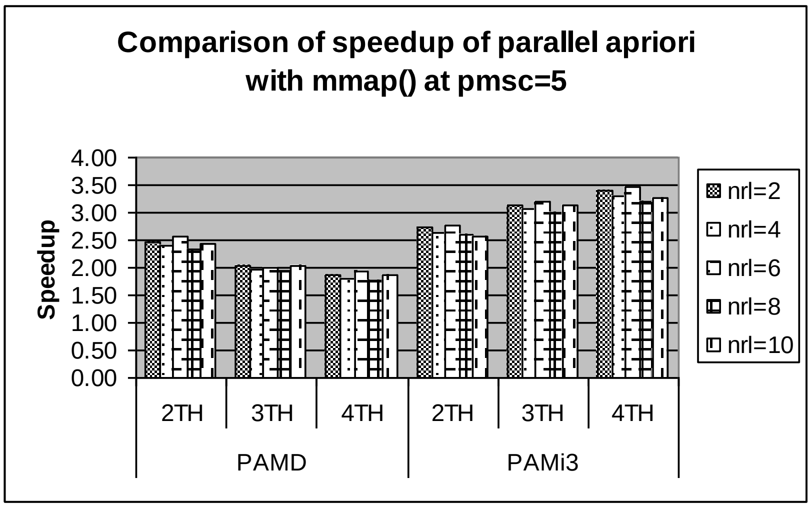


Fig 18: Comparison of Speedup values of parallel apriori with mmap() on Pentium DualCore Vs Corei3 at pmsc=5

VII. CONCLUSIONS

Though the computer hardware industry is introducing new processor architectures, a software application can take the performance gains from these architectures only when it is properly coded. Especially the architectures like multi-core and hyperthreading technologies can give the performance benefit to the application only when the application is properly written to run with multiple threads. In this paper the performance results of parallelizing the popular data mining algorithm called apriori using OpenMP threads on a hyperthreaded Intel Corei3 processor were presented and the results were compared with the results obtained a non hyperthreaded Intel Pentium DualCore processor . The performance of the algorithm was compared on these architectures with two different I/O retrieval techniques - conventional I/O technique using fread() and a special I/O technique using mmap(). Our results proved that the efficiency and speed up of our parallel implementation were more with a hyper threaded processor compared to non hyper threaded processor with both fread() and mmap() for all the datasets taken and at all the support counts considered.

REFERENCES

[1] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism", In Proc. of Int'l Symp. on Computer Architecture, pp. 392-403, Jun. 1995.

[2] Deborah T. Marr, Frank Binns, David L. Hill, Glenn Hinton, David A. Koufaty, J. Alan Miller, Michael Upton Hyper-Threading Technology Architecture and Microarchitecture. Intel Technology Journal, Volume 6, I ssue 1, February 14, 2002

[3] The Free Lunch Is Over A Fundamental Turn Toward Concurrency in Software By Herb Sutter [This article appeared in Dr.Dobb's Journal ,30(3),March 2005] Copyright © 2009 Herb Sutter

[4] Sarita V. Adve, Vikram S. Adve, gul Agha,Matthew I. Frank,María Jesús garzarán,John C. Hart, . . ., Craig Zilles (November 2008). "Parallel Computing Research at Illinois: The UPCRC Agenda" (PDF). Parallel@Illinois, University of Illinois at Urbana-Champaign)

[5] Multi-core Processor Wikipedia [Available] en.wikipedia.org/wiki/Multi-c**ore**_processor

[6] PARALLELISM VIA MULTITHREADED AND MULTICORE CPUS Angela C. Sodan, Jacob Machina, Arash Deshmeh, Kevin Macnaughton, and Bryan Esbaugh, *University of Windsor, Canada*

[7] The mystery of the scalability of a CPUintensive application on an Intel multi-core processor with HyperThreading (HT). By Mark Friedman http://performancebydesign.blogspot.co.uk/

[8] Performance Insights to Intel® Hyper-Threading Technology Garrett Drysdale, Antonio C. Valle, Matt Gillespie Submitted by Antonio Valles on Fri, 11/20/2009 software.intel.com › Home › Microsoft Windows* 8

[9] Mattwandel, Markus, Devyani Deuskar, John Holm, Herbert Mayer, Toby Opferman, and Ryan Thompson. 2009 "Performance Gains on Intel® Multi-Core, Multi-Threaded Core™ i7. Published at ICS'09 © Intel Corporation"

[10] Anuradha.T, Satya Pasad R and S N Tirumalarao. Parallelizing Apriori on Dual Core using OpenMP. *International Journal of Computer Applications* 43(24):33-39, April 2012. Published by Foundation of Computer Science, New York, USA.

[11] Anuradha.T, Satya Pasad R and S N Tirumalarao. Performance Evaluation of Apriori on Dual Core with Multiple Threads. *International Journal of Computer Applications* 50(16):9-16, July 2012. Published by Foundation of Computer Science, New York, USA

[12] Anuradha.T, Dr.Satya Prasad.R, Dr.Tirumala Rao.S.N Performance evaluation of apriori with memory mapped files. International Journal Of Computer Science Issues Vol.10, Issue 1,no1,January 2003

[13] *Introduction to PC Hardware* -- H. Gilbert *1 Jan, 2007* Copyright 1998, 2007 PCLT

[14] Intel® Hyper-Threading Technology Technical user's guide Copyright © 2003, Intel Corporation

[15] Hyper-Threading Technology, Multi-core, and Mobile Intel® Pentium® Processor-M Toolbox Submitted by Khang Nguyen (Intel) on Fri, 04/30/2010 software.intel.com › Home › Parallel Computing

[16] Shameem Akhter and Roberts Multicore programming increasing performance through software multithreading www.intel.com/intelpress Copyright © 2006 Intel Corporation

[17] TMurgent Technologies Hyper--Threading and Multiprocessor System Performance ON SERVER 2003 Should you enable Hyper-Threading? WHITEPAPER June 25,2003

[18] National Instruments India Understanding Parallel Hardware: Multiprocessors, Hyperthreading, Dual-Core, Multicore and FPGAs - Multicore Programming Fundamentals Whitepaper Series Publish Date: Dec 06, 2011

[19] J. Boisseau, K. Milfeld, and C. Guiang. “Exploring the Effects of Hyperthreading on Scientific Applications,” presented in Technical session number 7B, 45th Cray User Group Conference, Columbus, Ohio, May 2003

[20] W. Huang, J. Lin, Z. Zhang, and J. M. Chang. “Performance Characterization of Java Applications on SMT Processors,” International Symp. on Performance Analysis of Systems and Software (ISPASS), March 2005

[21] S. Blackburn, P. Cheng, and K. McKinley. “Myths and Realities: The Performance Impact of Garbage Collection,” Proc. SIGMETRICS ’04, June 2004

[22] Saini, Subhash, Haoqiang Jin, Robert Hood, David Barker, Piyush Mehrotra, and Rupak Biswas. "The impact of Hyper-Threading on processor resource utilization in production applications." In *High Performance Computing (HiPC), 2011 18th International Conference on*, pp. 1-10. IEEE, 2011

[23] Gasmi, Ghada, Amira Beji, Yahya Slimani, and Lotfi Lakhal. "Functional dependency mining: harnessing multicore systems." In Computer and Information Science (ICIS), 2011 IEEE/ACIS 10th International Conference pp. 135-139. IEEE, 2011.

[24] Tian, Xinmin, Yen-Kuang Chen, Milind Girkar, Steven Ge, Rainer Lienhart, and Sanjiv Shah. "Exploring the use of Hyper-Threading technology for multimedia applications with Intel® OpenMP compiler." In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pp. 8-pp. IEEE, 2003.

[25] Agrawal R, Srikant R “Fast algorithms for mining association rules” In: Proceedings of the 1994 international conference on very large data bases (VLDB’94), 1994 Santiago, Chile, pp 487–499

[26] Chap ter12 “Shared memory Introduction” www.kohala.com/start/unpv22e/unpv22e.chap12.pdf

[27] “OpenMP Architecture, OpenMP C and C++ Application Program Interface”, Copyright © 1997-2002 OpenMP Architecture Review Board [Available]http://www.openmp.org/.

[28] Ruud van der pas, “An Overview of OpenMP”, NTU Talk January 14 2009

[29] K Geurts, G Wets, T. Brijs and K. Vanhoof, “Profiling High Frequency Accident Locations Using Association Rules”, Electronic Proceedings of the 82th Annual Meeting of the Transportation Research Board, Washington, January 12-16, USA, 2003,18p.

[30] S N sivanandam, S Sumathi 2006 ―DataMining concepts,tasks and Techniques‖ First print 2006 by Thomson Business Information Pvt. Ltd., India.