# CPU Efficiency Enhancement through Offload

Naeem Akhter
Assistant Comp. Programmar,
University of Sargodha
Sargodha, Pakistan

Iqra Sattar
Lecturer, University of Lahore,
Sargodha Campus,
Sargodha, Pakistan

Furqan-ur-Rehman
Assistant Network Admin.
UCA, University of Sargodha,
Sargodha, Pakistan

## ABSTRACT
There are several causes of slowness in personal computers. While working on a PC to regularly execute jobs of similar nature, it is essential to be aware of the reasons of slowness to achieve the optimal CPU speed. A CPU being the most important unit of the whole system must be offloaded from unnecessary jobs which are not required at that moment. The checklist for ensuring the optimal PC speed is a long one but the paper discusses some major factors of PC slowness. It is empirically discussed how a PC user can offload the CPU to achieve optimal CPU speed. It is quantitatively proved that offloading can speedup the CPU upto a significant level.

## Keywords
Bubble Sort, Selection Sort, Gadgets, Offloading, Concurrent execution.

## 1. INTRODUCTION
It is factually evident that CPUs have the tendency to become more efficient in terms of speed and getting cheaper in terms of cost at an amazing rate[1]. It was first observed in 1965 by Gordon Moore, and is known as Moore's Law. Computers in the past were very slow in speed[2]. Only one program could have been run on those computers. But today's Operating Systems can manage execution of multiple programs concurrently. While working on a single-user computer with Windows Operating System various activities can be performed concurrently like, Installing a program, Creating or copying files, Downloading Programs, Printing various documents, Listening music, Running anti-virus program in the background and Typing texts etc.

CPU is the most important component of a whole computer system. Its speed is decreasing in parallel with the increasing number of concurrent processes. For example, a large number of tabs open in a browser, few of them may be auto-refreshing or live updates i.e. live news, cricinfo.com, weather updates etc. slows down CPU speed. Similarly, if there are a large number of add-ons in the browser, they always start popping or downloading at the opening of the browser, also suck CPU speed. Running too many applications at a time, takes a lot of memory, the resultant swapping also consumes a bit of CPU ticks. An Anti-virus program configured to run scans in the background is another source of keeping CPU under burden. Some rogue programs i.e heavy-duty videos also eat up CPU speed after encountering an error. Having too many programs at start-up also decreases the CPU speed, because all of them try to run when we start our computer. Another cause of low CPU speed is the use of too many gadgets at the Windows desktop i.e. Calendar, Picture Puzzle, Clock, Windows Media etc., Slide Show, Feed Headlines and Weather.

If a set of jobs with similar nature and computational requirement is required to execute on a single-user computer repeatedly, it will be an obvious need to enhance or maximize CPU speed. One way to enhance CPU speed is by disabling few operating system services which are not required for the solution of problem(s) in-hand. We can disable unwanted Operating System services from "*System Configurations*". The aim of the research is to analyze CPU speed whether it can be optimized by offloading from unnessary applications which share CPU ticks with computational jobs during execution. Difference of CPU speed has been calculated mathematically and the results have been displayed in tables and graphs. Rest of the paper comprises of: 2. Related Work 3. Methodology 4. Experiments 5. Results and Discussion 6. Conclusion 7. Future Work 8. References.

## 2. RELATED WORK
In [3] Sumit Basu etc have identified the various causes of slowness in personal computers. They have analyzed that slowness in personal computers may occur due to CPU, memory, IO, Handle Count, Page Faults and Thread Count. But one of the major causes of slowness is "One of the top two processes" out of the multiple processes running on a personal computer. It was observed in 28 cases out of the total 31 cases analyzed. They also concluded that the top process caused slowness in 26 out of 31 cases. In [4] it is mentioned that "Most determining factor in the speed of a PC is how much weight it carries". Moreover:

> "More than 60% of application programs most PC user install on a Personal Computer firstly, also install considerable number of unnecessary background tasks which suck up process and memory resources, sometimes so badly that a PC grinds to a half as a result. For example, the PC on which this document was written has 32 background tasks that I disabled because they either were totally unnecessary by any criteria, or they provide features I will personally never ever use. 32! Imagine if all those 32 programs were running right now, all sucking up CPU time and Memory space."

In [5] D. Cotroneo etc have stated that concurrent execution of multiple processes on a PC causes system hang. There are two ways aspects of system hang, processes waiting for resources for an unlimited time and processes encountering infinite loops. The researchers have proposed framework which avoids system hang by self-handling methodology. It improved performance overhead upto 0.6%.

## 3. METHODOLOGY
Research is a systematic and logical pursuit for new and meaningful information on a particular subject[6]. CPU Speed calculation is undoutebdly, a machine dependent assignment. It is compulsory duty of the researcher to explicitly describe the formal description of the system used for the research purpose. It will not only eliminate the question marks on the research but also provide the researchers an easy way to reverify the research results. Following is the data set and jobs on which experiments have been conducted.

## 3.1 Dataset

Three integer arrays of 1000, 1500 and 2000 sorted in descending order used for the following three jobs one by one, i.e. firstly, the following three jobs were executed on array of 1000 size, secondly, 1500 size and thirdly on the array with the size of 2000 integer.

***3.1.1*** ***Job 1:*** Sort using Bubble Sort Algorithm.

***3.1.2*** ***Job 2:*** Sort using Selection Sort Algorithm.

***3.1.3*** ***Job 3:*** Multiply each item of the array by 2.

The above mentioned three jobs were run on the machine in the following situations separately and difference has been calculated.

***3.1.4*** ***With Load:*** Executing the jobs concurrently with the following:

*3.1.4.1* 2 instances of VLC player each running video media file (.mp4) of 50 minutes duration.

*3.1.4.2* Windows Gadgets (Clock, Calendar, CPU Meter, Feed Headline, Currency, Slide Show, Weather, Picture Puzzle, Windows Media) running on the desktop.)

***3.1.5*** ***With Offload:*** Running all the jobs after closing all windows gadgets and both VLC player instances (offloading the CPU).

A program was developed in C# to calculate CPU time consumed by all three jobs, separately on the above mentioned data (1000, 1500 and 2000 integers) in both *'with load'* and '*off load'* scenarios. Operating system used for the experiments is Windows 7 Service Pack 1, (64 bit), formal description of the machine is as under:

- Core(TM)2 Duo, Intel(R)
- T7100 @ 1.80GHz (CPU)
- RAM 2.00 GB

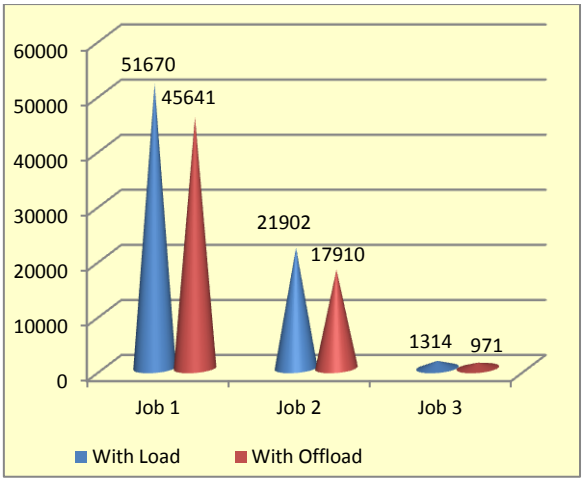
The results were calculated, tabulated and presented in the form of graphs using MS Excel version 2010.

## 4. EXPERIMENTS & RESULTS

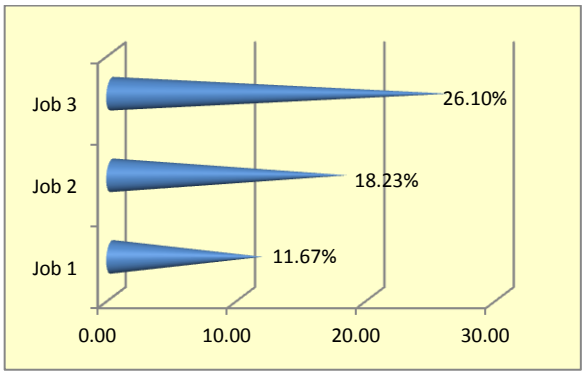
To achieve high accuracy in calculating CPU time of specific piece of code '*Stopwatch'* can be used. But 100% accuracy is not guaranteed. Thomas Maierhofer **[7]** states that result of *Stopwatch* may be 25%-30% different when we repeatedly execute the same code on the same computer. To achieve more accuracy, program was executed five (5) times and average of the five (5) outputs have been calculated. Pankaj Sareen in [8] has also done the same in his sorting algorightms comparison.

**Table – 1 Execution of Jobs on 1000 integers**

| Details | CPU Time Consumption in Microseconds | | | | | |
|---|---|---|---|---|---|---|
| | Job 1 | | Job 2 | | Job 3 | |
| | With Load | With Off-Load | With Load | With Off-Load | With Load | With Off-Load |
| 1st Run | 51312 | 47187 | 22288 | 17522 | 1208 | 1046 |
| 2nd Run | 50787 | 48943 | 22156 | 19012 | 1632 | 1007 |
| 3rd Run | 50950 | 38863 | 19575 | 21807 | 1136 | 1142 |
| 4th Run | 52524 | 45582 | 24306 | 15521 | 1420 | 749 |
| 5th Run | 52776 | 47629 | 21184 | 15687 | 1174 | 911 |
| **Avg** | **51670** | **45641** | **21902** | **17910** | **1314** | **971** |
| **Diff.** | **6029** | | **3992** | | **343** | |
| **Saving** | **11.67%** | | **18.23%** | | **26.10%** | |




**Graph – 1.1 Execution of Jobs on 1000 integers**

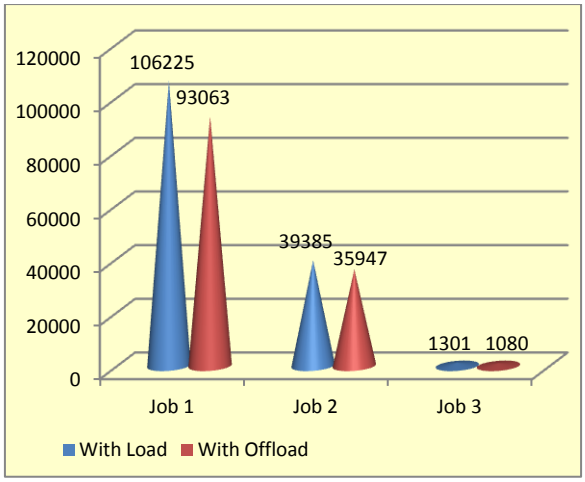



**Graph – 1.2 CPU Speed Enhancement on 1000 integers**

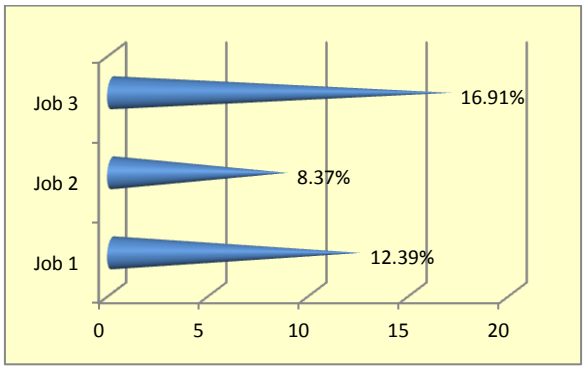
On input 1000 integers, in the first job CPU time saving of 6929 microsecond(11.67%), second job 3992 (18.23%) and third job 343 microseconds (26.10%) was noted.

**Table – 2 Execution of Jobs on 1500 integers**

| Details | CPU Time Consumption in Microseconds | | | | | |
|---|---|---|---|---|---|---|
| | Job 1 | | Job 2 | | Job 3 | |
| | With Load | With Off-Load | With Load | With Off-Load | With Load | With Off-Load |
| 1st Run | 115365 | 95899 | 35936 | 39149 | 1632 | 1078 |
| 2nd Run | 109325 | 88261 | 38553 | 33716 | 1260 | 1054 |
| 3rd Run | 107408 | 91902 | 50356 | 38102 | 1133 | 1087 |
| 4th Run | 97579 | 89916 | 33760 | 34055 | 1385 | 1103 |
| 5th Run | 101446 | 99335 | 38319 | 34713 | 1093 | 1079 |
| **Avg** | **106225** | **93063** | **39385** | **35947** | **1301** | **1080** |
| **Diff.** | **13162** | | **3438** | | **220** | |
| **Saving** | **12.39%** | | **8.73%** | | **16.91%** | |




**Graph – 2.1 Execution of Jobs on 1500 integers**

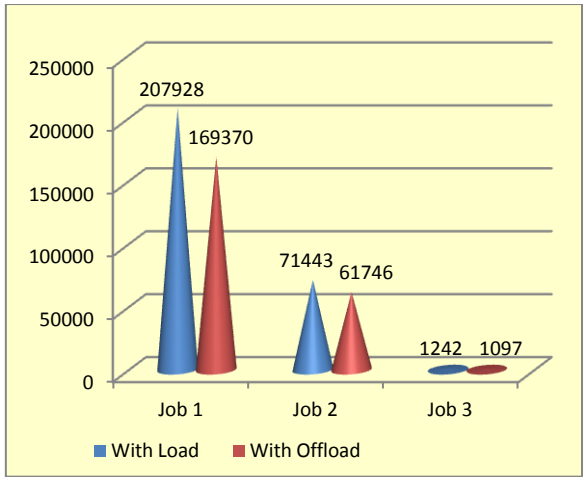



**Graph – 2.2CPU Speed Enhancement on 1500 integers**

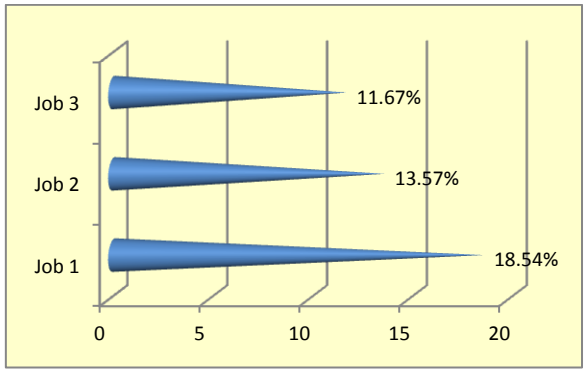
On input 1500 integers, in the first job CPU time saving of 13162 microsecond(12.39%), second job 3438 (8.73%) and third job 220 microseconds (16.91%) was noted.

**Table – 3 Execution of Jobs on 2000 integers**

| Details | CPU Time Consumption in Microseconds | | | | | |
|---|---|---|---|---|---|---|
| | Job 1 | | Job 2 | | Job 3 | |
| | With Load | With Off-Load | With Load | With Off-Load | With Load | With Off-Load |
| 1st Run | 205920 | 173142 | 72197 | 60949 | 1148 | 1112 |
| 2nd Run | 198860 | 180103 | 71057 | 65942 | 1440 | 1129 |
| 3rd Run | 208665 | 172898 | 78356 | 60780 | 1203 | 1075 |
| 4th Run | 217854 | 160497 | 65852 | 62768 | 1191 | 1095 |
| 5th Run | 208341 | 160208 | 69755 | 58289 | 1228 | 1074 |
| **Avg** | **207928** | **169370** | **71443** | **61746** | **1242** | **1097** |
| **Diff.** | **38558** | | **9698** | | **145** | |
| **Saving** | **18.54%** | | **13.57%** | | **11.67%** | |




**Graph – 3.1 Execution of Jobs on 2000 integers**

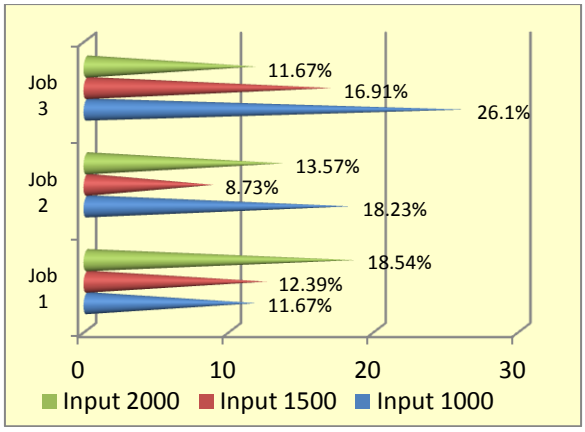



**Graph – 3.2 CPU Speed Enhancement on 2000 integers**

On input 2000 integers, in the first job CPU time saving of 38558 microsecond(18.54%), second job 9698 (13.57%) and third Job 145 microseconds (11.67%) was noted.

**Table – 4 CPU Time Savings of all jobs on all inputs**

| Input | CPU Time Consumption in Microseconds | | | | | |
|---|---|---|---|---|---|---|
| | Job 1 | | Job 2 | | Job 3 | |
| | With Load | With Off-Load | With Load | With Off-Load | With Load | With Off-Load |
| 1000 | 51670 | 45641 | 21902 | 17910 | 1314 | 971 |
| 1500 | 106225 | 93063 | 39385 | 35947 | 1301 | 1080 |
| 2000 | 207928 | 169370 | 71443 | 61746 | 1242 | 1097 |
| **Total** | **365822** | **308073** | **132730** | **115602** | **3857** | **3148** |
| **Diff.** | **57749** | | **17128** | | **708** | |
| **Saving** | **15.79%** | | **12.90%** | | **18.36%** | |




**Graph – 4 CPU Time Savings of all jobs on all inputs**

## 5. DISCUSSION

Three jobs were defined and experiments were conducted on three input (1000, 1500 and 2000 integers) values. First job is performance of Bubble Sort. Its performance in terms of CPU time consumption decreases with the increase of input size. The results of first job show that the saving of CPU time increased gradually with the increase in input value i.e.

| | | |
|---|---|---|
| Input 1000 | = | 11.67% |
| Input 1500 | = | 12.39% |
| Input 2000 | = | 18.54% |

From these result it can be estimated that with the further increase of input value i.e. 5000 or 8000 integers the saving of CPU time will increase accordingly. A clear pattern can be noted that as biger the input as much the saving of CPU. If we analyze the CPU time saving of second job, the following savings were noted:

| | | |
|---|---|---|
| Input 1000 | = | 18.23% |
| Input 1500 | = | 8.73% |
| Input 2000 | = | 13.57% |

CPU time saving is noted on all three inputs but there is no regularity or pattern as was found in the case of first job. It is due to the nature of Selection Sort algorithm used for the second job. It behaves differently on various input sizes in terms of CPU time consumption. As Jehad Hammad in [9] discusses in his comparative study on HornerEval, Linear Search, Towers, Binary Search, Insertion, Max, Min, MaxMin, Merge, Quick, SelectionSort, Heap, Bubble and Gnome Sorting algorithms on 5000, 10000, 20000 and 30000 input values that Selection sort is quicker than bubble sort and gnome sort. He has further analyzed a drawback of selection sort which continues sorting the items if they are already arranged, while gnome and bubble sort algorithms swap the items if required. Results in terms of CPU time saving of third job are opposite to first job:

| | | |
|---|---|---|
| Input 1000 | = | 26.10% |
| Input 1500 | = | 16.19% |
| Input 2000 | = | 11.67% |

This job is of very short size as compared to other two jobs. In the case of job 3, the CPU time saving decreased with the increasing input value. Summing up the discussion it can be concluded that in job1 the dominant factor is the job itself. For job2 the dominant factors were both 'the load' and job itself. But in case of job3, the dominant factors was 'the load' only.

## 6. CONCLUSION

Offloading the CPU from unnecessary programs increases CPU efficiency and saves CPU time upto significant amount of time. The amount of time saved depends upon the nature and size of jobs to be executed i.e. the dominant factor may be the computation jobs or may be the 'burden' (irrelevant programs) running concurrently or both the 'jobs' and the 'load'. The most important outcome of this research is the quantitative measures which indicate how a PC user can improve the efficiency of CPU or save CPU energy waisted by the 'load'. The message to the PC users is clear that they should offload the CPU by closing all the programs and disable all the windows services not required during the execution of computation jobs in hand to achieve optimized CPU efficiency.

## 7. FUTURE WORK

i. In future, experiment on more CPU offload scenarios can be conducted to recommend CPU efficiency enhancements for PC users in varying perspectives.

ii. CPU time consumption of major Windows Services can be calculated separately running concurrently with a task set.

iii. Experiment can be conducted usig variety of machines in order to calculate CPU efficiency.

## 8. REFERENCES

[1] Douglas M. Pase and Matthew A. Eckl A "Comparison of Single-Core and Dual-Core Opteron" , IBM xSeries Performance Development and Analysis, 3039 Cornwallis Rd., Research Triangle Park, NC 27709-2195.

[2] Omer, M.A., Zwaid, M. J. (2011) "CPU Scheduling" A Project submitted to Republic of Iraq, Scientific Research, University of Baghdad, College of Science, Department of Computer Sciences.

[3] Basu, S., Dunagun, J., Smith, G. (2011), "Why Did My PC Suddenly Slow Down?", Microsoft Research, One Microsoft Way, Redmond, WA 98052.

[4] "Speed Up your PC – The Ultimate Guide to drastically Improving Your PC Speed & Performance - *The Science of Improving PC Speed*", AnswersThatWork.com, 26-March-2011.

[5] D. Cotroneo, R. Natella, S. Russo. (2009) "Assessment and improvement of hang detection in the Linux operating system," In SRDS, New York, USA, pp. 288-294.

[6] Rajasekar, S., Philominathan, P., Chinnathambi, V. (2013), "Research Methodology", arXiv.org > physics > arXiv: physics/0601009v3.

[7] Thomas Maierhofer Performance Tests: Precise Run Time Measurements with System.Diagnostics.Stopwatch. (2010),

[8] Sareen, P. Comparison of Sorting Algorithms (On the Basis of Average Time), International Journal of Advanced Research in Computer Science & Software Engineering, Vol. 3, Issue 3 (2013).

[9] Hammad, J. A Comparative Study between Various Sorting Algorithms, International Journal of Computer Science and Network Security (IJCSNS), Vol 15, No. 3 (2015).