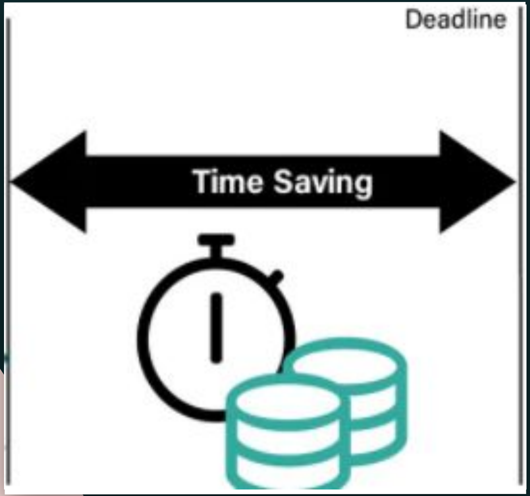
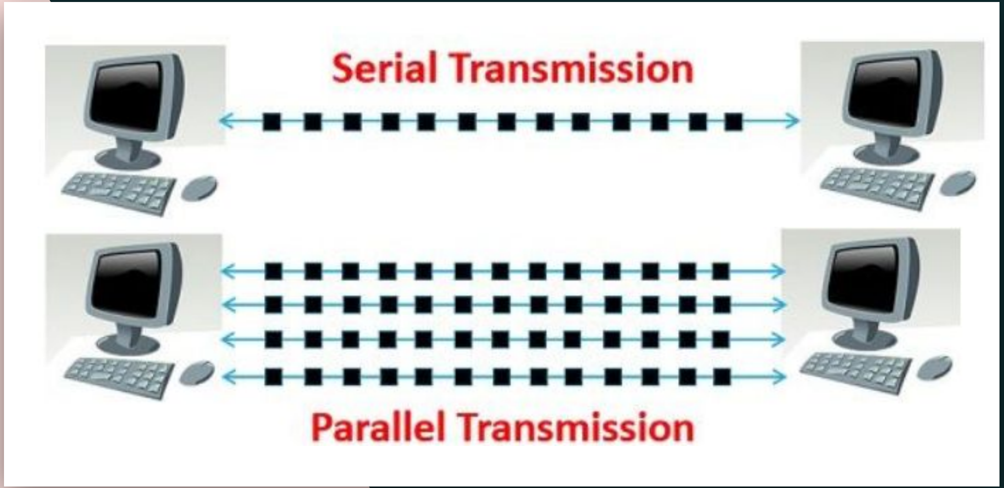
ADA(CO-208)

Parallel vs Serial Computing





Submitted By:
Abhishek Kumar Singh (2K19/CO/021)

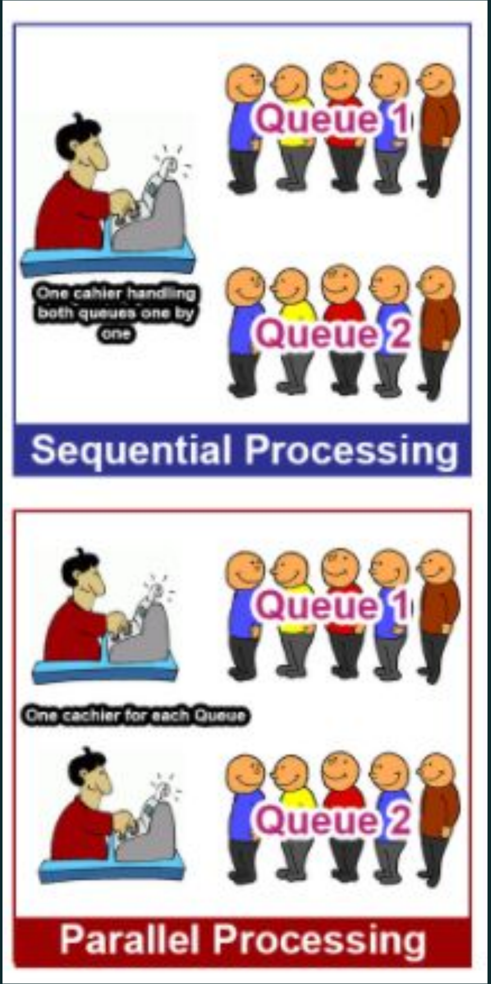
Amit Kumar Jha (2K19/CO/053)

Submitted To:

Mr. Sanjay Kumar

PROBLEM STATEMENT

In computers, serial processing is the processing of program instructions just one at a time whereas parallel processing is the processing of program instructions by dividing the program into multiple fragments and processing these fragments simultaneously. The objective of parallel processing is running a program in less time. Parallel computing has been around for many years. Recently, the interest in parallel computing has grown due to the introduction of multi core processors at a reasonable price for the common people. The goal of this paper is to analyze and compare serial computing with parallel computing.


Queue 1
Queue 2
One cahier handling both queues one by one
Sequential Processing
Queue 1
One cachier for each Queue
Queue 2
Parallel Processing

INTRODUCTION

Serial Computing

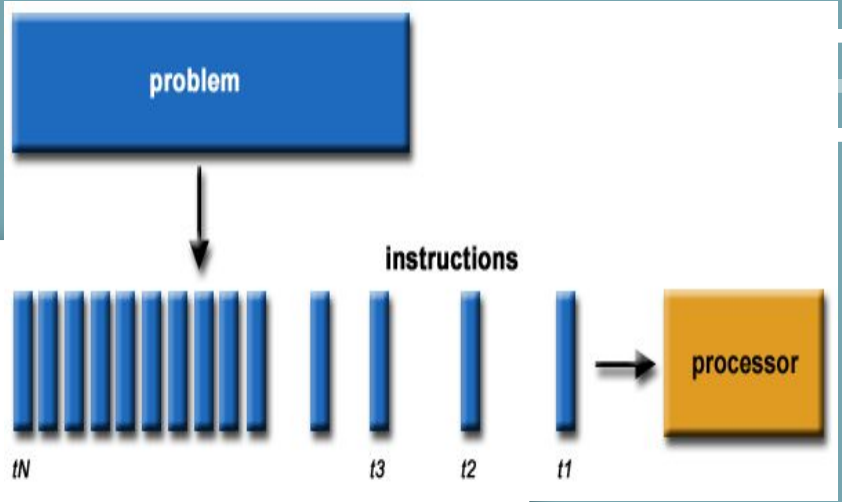
- ➔ Serial computing is a type of computation where instructions are executed sequentially one after another
- ➔ In serial computing, a problem is broken into a series of instructions and the instructions are then executed sequentially one after another. Only one instruction may execute at any moment in time.
- ➔ A serial processor is a processor type used by systems where the central processing unit (CPU) carries out just one machine-level operation at a time.
- ➔ In 2005 Intel launched the first dual-core processor for end users; prior to that, every computer processor used the serial processing technology.

Parallel Computing

- ➔ Parallel computing is a type of computation where many calculations or the execution of processes are carried out simultaneously.
- ➔ Large problems can often be divided into smaller ones, which can then be solved at the same time. There are several different forms of parallel computing: bit-level, instruction-level, data, and task parallelism.
- ➔ Parallelism has long been employed in high-performance computing, but has gained broader interest due to the physical constraints preventing frequency scaling.
- ➔ As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multi-core processors.

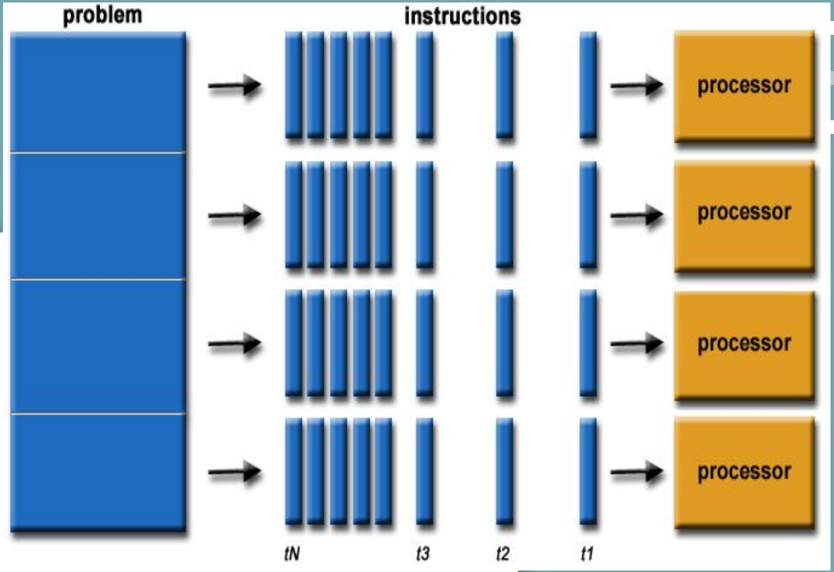
- ➔ Today, new applications arise and demand faster computers. Commercial applications are the most used on parallel computers.
- ➔ A computer that runs such an application; should be able to process large amounts of data in sophisticated ways.
- ➔ These applications include graphics, virtual reality, and decision support, parallel databases, medicine diagnosis and so on.
- ➔ We can say with no doubt that commercial applications will define future parallel computers architecture but scientific applications will remain important users of parallel computing technology.

Limits to serial computing



- ❑ Transmission speeds - the speed of a serial computer is directly dependent upon how fast data can move through hardware. Absolute limits are the speed of light (30 cm/nanosecond) and the transmission limit of copper wire (9 cm/nanosecond). Increasing speeds necessitate increasing proximity of processing elements.
- ❑ Limits to miniaturization - processor technology is allowing an increasing number of transistors to be placed on a chip. However, even with molecular or atomic-level components, a limit will be reached on how small components can be.
- ❑ Economic limitations - it is increasingly expensive to make a single processor faster. Using a larger number of moderately fast commodity processors to achieve the same (or better) performance is less expensive.

Advantages of Parallel computing



- ❑ Save time and/or money: In theory, throwing more resources at a task will shorten its time to completion, with potential cost savings. Parallel computers can be built from cheap, commodity components.
- ❑ Solve larger problems: Many problems are so large and/or complex that it is impractical or impossible to solve them on a single computer, especially given limited computer memory.
- ❑ Provide concurrency: A single compute resource can only do one thing at a time. Multiple computing resources can be doing many things simultaneously

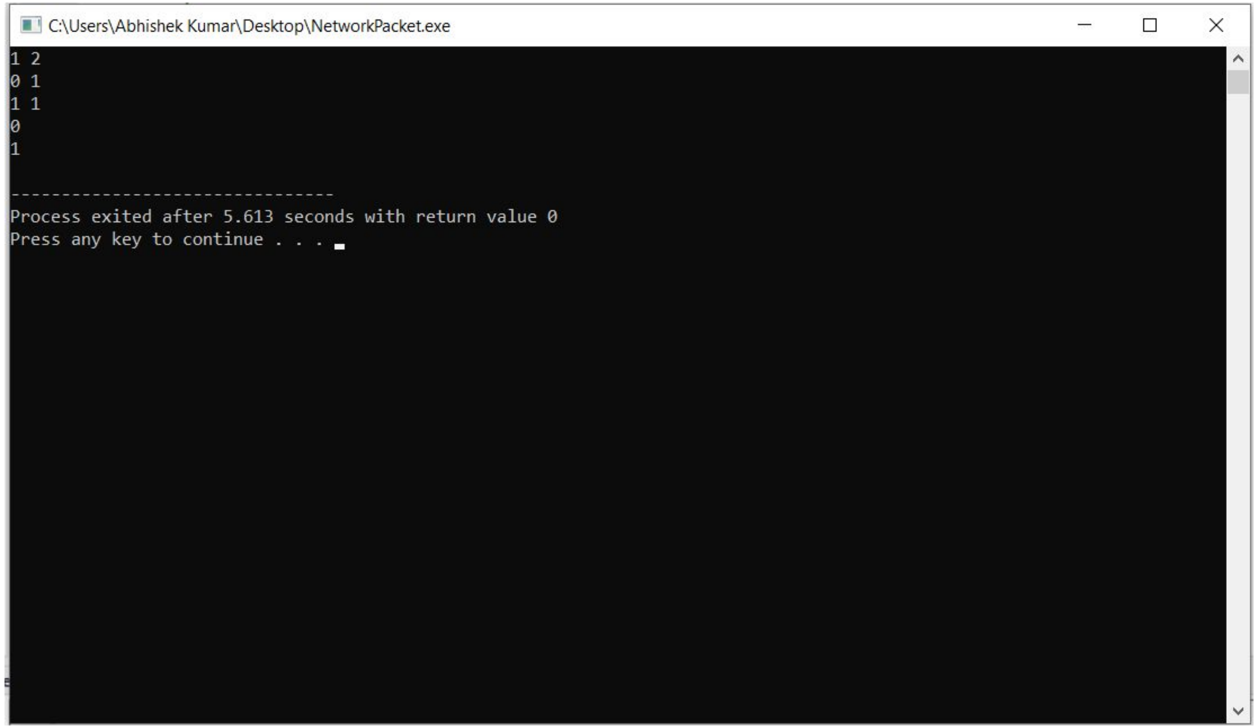
RESULTS

Serial Computing

Input : The first line of the input contains the size S of the buffer and the number n of incoming jobs. Each of the next n lines contains two numbers. i-th line contains the time of arrival Ai and the processing time Pi (both in milliseconds) of the i-th job. It is guaranteed that the sequence of arrival times is non-decreasing (however, it can contain the exact same times of arrival in milliseconds — in this case the job which is earlier in the input is considered to have arrived earlier).

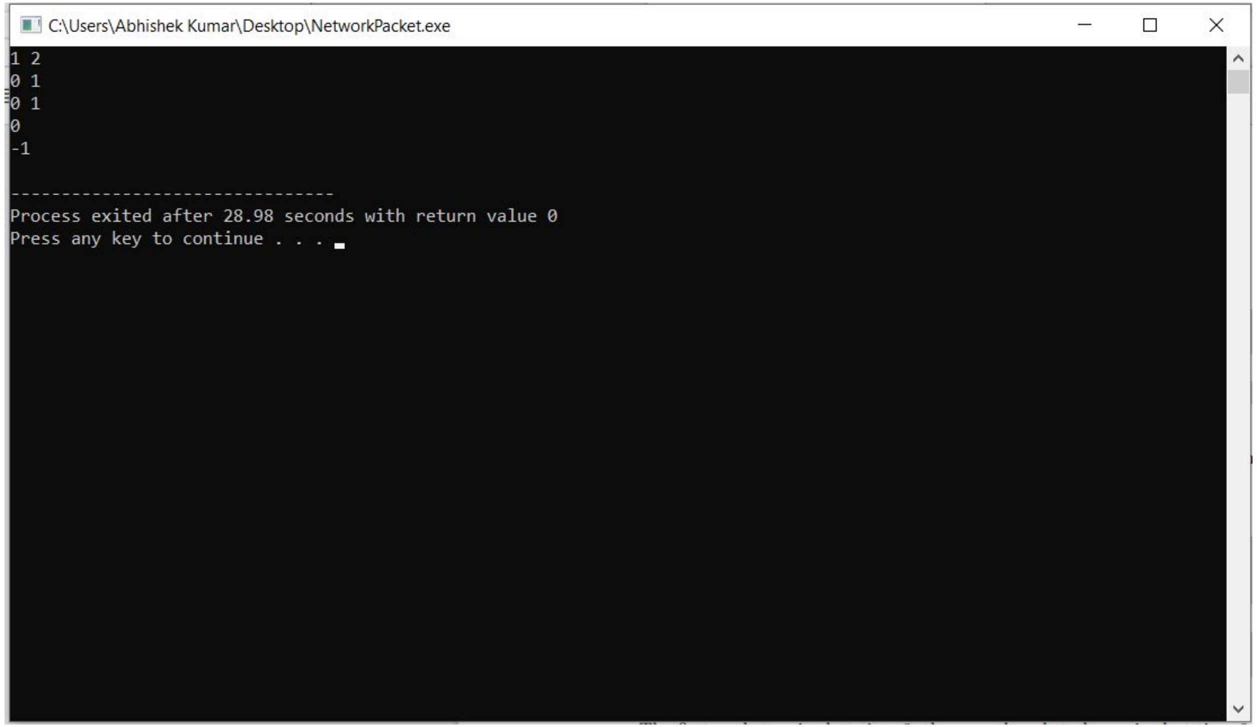
Output : For each job output either the moment of time (in milliseconds) when the processor began processing it or −1 if the job was dropped (output the answers for the jobs in the same order as the jobs are given in the input).

Explanation : The first job arrived at time 0, the computer started processing it immediately and finished at time 1. The second job arrived at time 1, and the computer started processing it immediately.


C:\Users\Abhishek Kumar\Desktop\NetworkPacket.exe

```
1 2
0 1
1 1
0
1

--------------------------------
Process exited after 5.613 seconds with return value 0
Press any key to continue . . .
```

Explanation : The first job arrived at time 0, the second job also arrived at time 0, but was dropped, because the buffer has size 1 and it was full with the first job already. The first job started processing at time 0, and the second job was not processed at all.



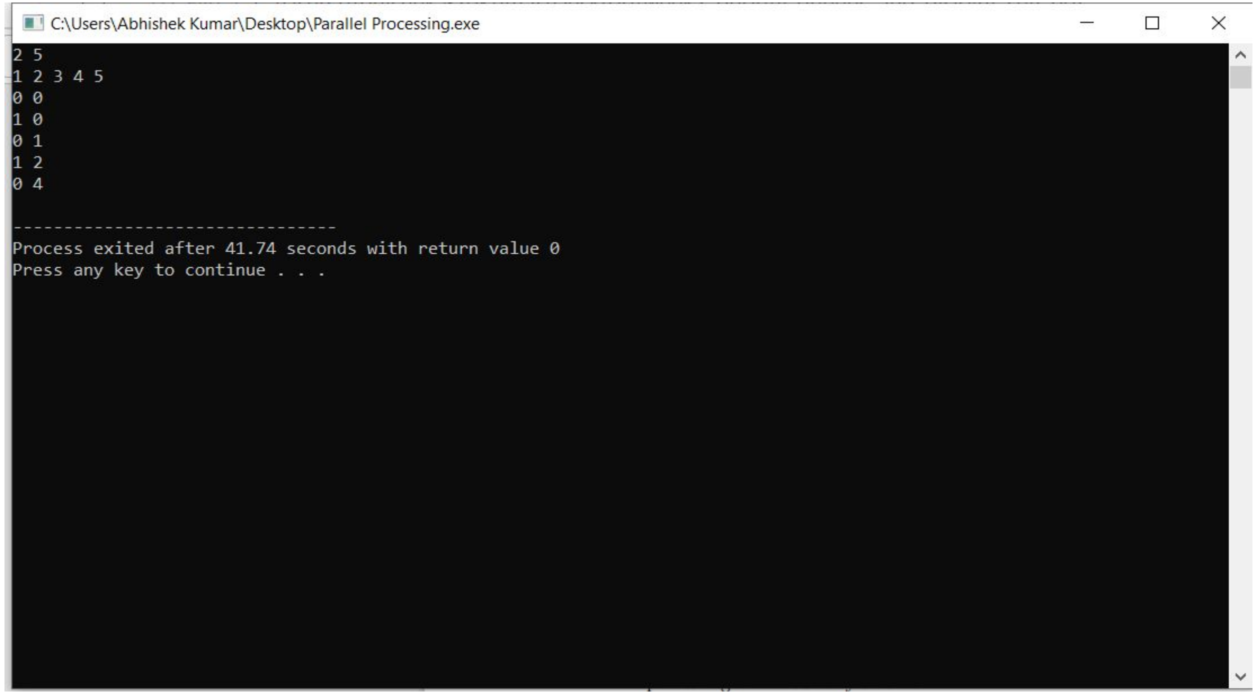
```
1 2
0 1
0 1
0
-1

--------------------------------
Process exited after 28.98 seconds with return value 0
Press any key to continue . . .
```

Parallel Computing

Input : The first line of the input contains integers n and m. The second line contains m integers ti — the times in seconds it takes any thread to process i-th job. The times are given in the same order as they are in the list from which threads take jobs. Threads are indexed starting from 0.

Output : Output exactly m lines. i-th line (0-based index is used) should contain two space separated integers — the 0-based index of the thread which will process the i-th job and the time in seconds when it will start processing that job.

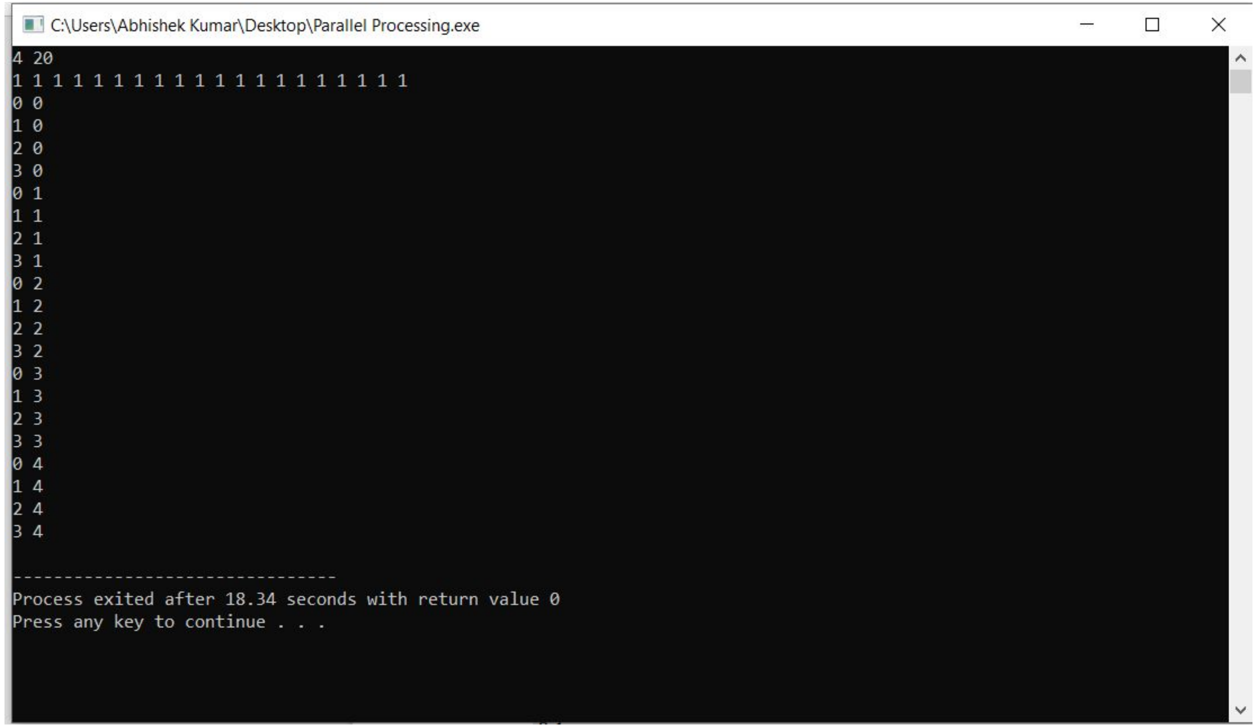
Explanation : The two threads try to simultaneously take jobs from the list, so thread with index 0 takes the first job and starts working on it at the moment 0. The thread with index 1 takes the second job and starts working on it also at the moment 0. After 1 second, thread 0 is done with the first job and takes the third job from the list, and starts processing it immediately at time 1. One second later, thread 1 is done with the second job and takes the fourth job from the list, and starts processing it immediately at time 2. Finally, after 2 more seconds, thread 0 is done with the third job and takes the fifth job from the list, and starts processing it immediately at time 4.



C:\Users\Abhishek Kumar\Desktop\Parallel Processing.exe

```
2 5
1 2 3 4 5
0 0
1 0
0 1
1 2
0 4

--------------------------------
Process exited after 41.74 seconds with return value 0
Press any key to continue . . .
```

Explanation : Jobs are taken by 4 threads in packs of 4, processed in 1 second, and then the next pack comes. This happens 5 times starting at moments 0, 1, 2, 3 and 4. After that all the 5 × 4 = 20 jobs are processed.


C:\Users\Abhishek Kumar\Desktop\Parallel Processing.exe

```
4 20
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0
1 0
2 0
3 0
0 1
1 1
2 1
3 1
0 2
1 2
2 2
3 2
0 3
1 3
2 3
3 3
0 4
1 4
2 4
3 4

--------------------------------
Process exited after 18.34 seconds with return value 0
Press any key to continue . . .
```

CONCLUSION:

We have successfully implemented and simulated the serial and parallel processing of jobs. The parallel processing saves time and money as many resources working together will reduce the time and cut potential costs. Real world data needs more dynamic simulation and modeling, and for achieving the same, parallel computing is the key. It can be impractical to solve larger problems on serial computing. Complex, large datasets, and their management can be organized only and only using parallel computing's approach. Serial computing 'wastes' the potential computing power, thus Parallel Computing makes better work of hardware.

REFERENCES:

1. Ms. Rupa Yashwanta Nagpure and Prof Sandhya Dahake, “Research Paper on Basic Parallel Processing”, IOSR Journal of Engineering (IOSR JEN), ISSN (e): 2250-3021, ISSN (p): 2278-8719, PP 77-83
2. Abhay B. Rathod, Rajratna Khadse and M Faruk Bagwan, “SERIAL COMPUTING vs. PARALLEL COMPUTING: A COMPARATIVE STUDY USING MATLAB”, International Journal of Computer Science and Mobile Computing, Vol. 3, Issue. 5, May 2014, pg.815 – 820