Multicore Parallel Processing Concepts for Effective Sorting and Searching

Kota Sujatha
Research Scholar: CSE Department
GITAM University
Visakhapatnam, India

A Arjuna Rao
Professor & Director
Miracle Educational Society Group of Institutions
Bhogapuram, Vizianagram, India

P V Nageswara Rao
Professor, CSE Department
GITAM University
Visakhapatnam, India

V G Sastry, V Praneeta, Raj Kumar Bharat
B.Tech. Students
Miracle Educational Society Group of Institutions
Bhogapuram, Vizianagram, India

***Abstract*—Many applications today require more computing power than offered by a traditional sequential computers. High performance computing requires parallel processing. Parallelism is utilized to depict executions that physically execute at the same time with the objective of taking care of a complex issue quicker. Multicore processing means code working on more than one core of a single CPU chip and is a subset of Parallel Processing and Multicore utilization means that efficient usage of CPU. In parallel processing, program instructions are divided among multiple processors with the goal of executing the same program in less time compared to sequential processing. Applications programs related to sorting and searching are developed encapsulating parallel processing and are tested on huge database. Experimental results based on bubble sort and linear search show that it is easier to get work done if load is shared and also quicker by parallel processing with multicore utilization compared to sequential processing.**

Keywords—parallel processing; Parallelism; Multicore processing; Multicore utilization; Sorting; Searching; Sequential Processing;

I. INTRODUCTION

As a rule, parallel preparing implies that no less than two microchips handle parts of a general errand. The complex issue is partitioned into segment parts and afterward doles out every part to a committed processor. Every processor comprehends its piece of the general computational issue. The result reassembles the information to achieve the end finish of the first intricate issue[1].

The concurrent utilization of more than one CPU or processor centre to execute a project or various computational strings is the ticket in Parallel transforming. Preferably, parallel preparing makes projects run quicker in light of the fact that there are more CPUs or cores running it[2]. With single-CPU, single-core machines, it is conceivable to perform parallel transforming by joining the machines in a system. Nonetheless, this sort of parallel handling obliges exceptionally modern programming called disseminated transforming programming[3].

In practice, it is frequently hard to separate a project in such a path, to the point that different CPUs or cores can execute distinctive bits without meddling with one another. Desktop CPU manufacturers shifted to multi-core processor architectures in the recent past to address the growing performance demands and the exponential growth of power consumption of single core processors. In order to demonstrate multicore utilization, the architecture of Quad core processor is considered as shown in figure 1.

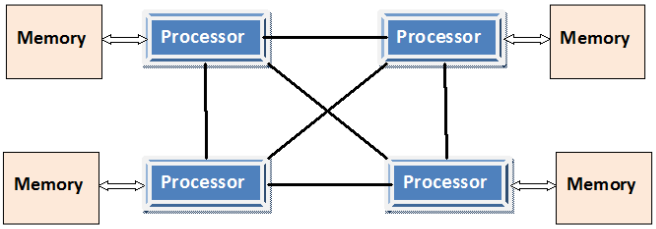

Fig. 1. Architecture of Quad Core Processor.

The advantages of Quad core processor are Multitasking, Run intensive applications simultaneously, less heat and power consumption, long term usage.

Multitasking: The quad-core framework is one of the best frameworks for multi-tasking. With such a variety of cores, it can do numerous techniques on the double, while as of now keeping up the honesty of the framework.

Run escalated applications: Applications that utilization up a great deal of assets, for example, illustrations programs, feature editors, and against infection projects, can run easily in the meantime.

Less hotness and force utilization: Most of the more up to date quad-core chips are so little and proficient; they can really

utilize less power and create less high temperature than single-core frameworks.

Use for long haul: The issue with Moore's Law is that it basically ensured that your machine would be old in around 24 months. Since few programming projects are modified to run on doublecore, significantly less quadcore, these processors are really route in front of programming improvement.

Hence by using multiple core CPUs of today can complete more work faster, and at lower power, than their single core predecessors. Most computers have just one CPU, but some models have several, and multi-core processor chips are becoming the norm. Hence it is easy to implement parallel processing with multi-core utilization on many problems by breaking them into modules that can be executed parallel on a multicore processor.

II. Parallel Processing with Multi-Core Utilization

For a few decades PC processors were essentially single core core architectures and CPU producers expanded execution by expanding core sizes, working frequencies, and utilizing littler assembling methodologies permitting more transistors in the same chip range. Notwithstanding, PC producers understood that the constant increment in recurrence and core sizes created exponential increments in force utilization and extreme high temperature dispersal. In this manner, CPU producers created multi-core CPU architectures to keep conveying higher execution processors, while restricting the force utilization of these processors. Most desktop and journal frameworks today utilize either a double or quad core processor and expend fundamentally lower power than their single core ancestors[5].

Parallel and multi-core preparing both allude to the same thing: the capacity to execute code in the meantime in more than one CPU/Core/machine. So multi-core intends to do parallel handling. A core is similar to a little processor inside a processor. So making code work for multicore preparing will dependably allude to parallelization.

As multicore architectures overwhelm single-core centered architectures in today's and future machine frameworks, customary applications with consecutive calculations can no more depend on engineering scaling to enhance execution. Rather, applications must switch to parallel calculations to take advantage of multicore framework execution. A multi-core centered construction modeling is characterized by the accompanying qualities:

- Architecture comprises of two or more indistinguishable CPU core.
- All cores impart a typical framework memory and are controlled by a solitary Operating framework.
- Each CPU is equipped for working freely on distinctive workloads and at whatever point conceivable, is additionally fit for imparting workloads

The structure depicting the process of parallel processing with multicore utilization on Quad core processor is shown in figure 2. Here the program is logically broken into four threads and each thread is executed by one core of the processor. As this is a Quad Core system four threads will run in parallel. The four results obtained are again combined and fed back to the program which again proceeds with further logic.

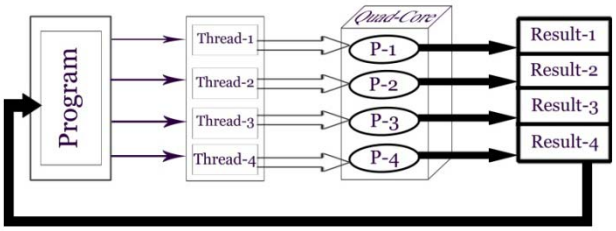

Fig. 2. Structure depicting the process of Parallel processsing with multicore utilization on Quad Core Processor.

As shown is Figure 2, multicore is a subset of parallel and concurrency can occur with or without parallelism. Here the task is divided into modules where each module will run as a thread on one processor. As the paper is based on results tested on quad core processor, four processors are shown on which one thread is being run.

Because of assignment sharing, the cores don't have to run at full limit and can be run at a lower recurrence and voltage. Since the force utilization of semiconductor gadgets is relative to the frequency and voltage-squared, even a little lessening in the working frequency and voltage will bring about critical diminishment in force utilization. This is demonstrated in Figure 3.. Here the same task is assigned to both single core and quad core and Quad core consumes less power for the same task that is executed on Single core because in Quad core each core is not utilized 100%.

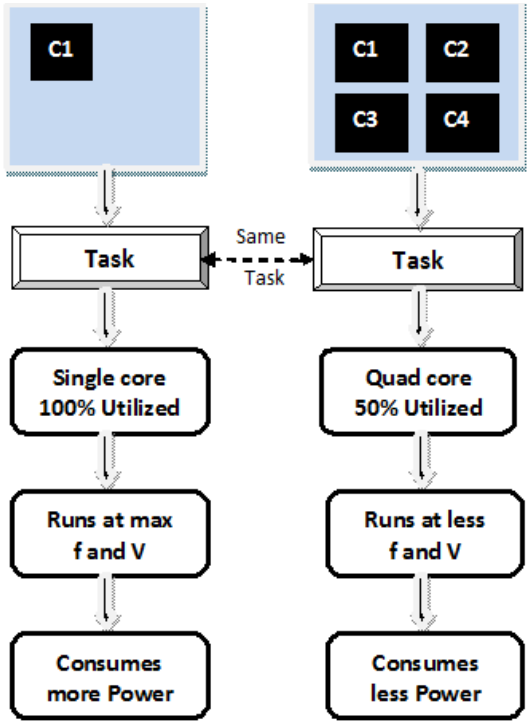

Fig. 3. Voltage Benefits of Quad Core CPU

Therefore a processor with a multicore CPU will often be more power efficient than a single core CPU based mobile processor. Hence programs developed based on multicore utilization run faster than single core CPU with better performance.

III. Results and Discussion

Multicore utilization can be applied to any problems that have huge data and processing involved. This is tested on two regular cases, first one on Sorting with Bubble Sort and next one on Searching with Linear Search. Two programs are developed for each problem one for single core and another one for quad core with same logic.

A. Case Study on Bubble Sort

In the air pocket sort, as components are sorted they step by step "air pocket" (or ascent) to their legitimate area in the exhibit, in the same way as air pockets climbing in a glass of pop. The air pocket sort more than once looks at contiguous components of a show. The main and second components are analyzed and swapped if out of request. At that point the second and third components are analyzed and swapped if out of request. This sorting procedure proceeds until the last two components of the exhibit are looked at and swapped if out of request. While asymptotically equivalent to the other $O(n^2)$ algorithms, it will require $O(n^2)$ swaps in the worst-case[6].

Bubble sort program is developed for both single core and quad core and the results are presented in Table 1.

TABLE I. BUBBLE SORT EXECUTION TIME CONSUMPTION (IN MILLI SECONDS) CHART

Trail No.	No of Values in Dataset							
	1000		3000		5000		7000	
	SC	QC	SC	QC	SC	QC	SC	QC
1	171	156	1201	1186	2699	2621	4587	2849
2	188	171	1185	1139	2761	2559	5132	5010
3	156	141	1373	1263	2667	2636	5226	5023
4	202	172	1248	1201	2652	2559	5272	5149
5	172	156	1248	1217	2683	2574	4665	4539
6	171	140	1264	1170	2746	2574	5148	4790
7	188	156	1240	1186	2574	2480	5008	4680
8	156	140	1279	1217	2684	2637	4430	4337
9	187	172	1326	1170	2559	2543	5241	5055
10	188	156	1124	1105	2512	2434	4680	4540

SC : Single Core QC: Quad Core

B. Case Study on Selection Sort

The thought of determination sort is that over and again the following biggest (or littlest) component in the cluster if discovered and move that it to its last position in the sorted show. To sort the cluster in expanding request, i.e. the most diminutive component towards the start of the cluster and the biggest component towards the end, first the biggest component is chosen and moved it to the most noteworthy record position. This is carried out by swapping the component at the most astounding file and the biggest component. At that point the successful size of the exhibit is lessened by one component and rehashes the procedure on the more modest (sub) array. The methodology stops when the powerful size of the cluster turns into 1[7].

The effectiveness of a calculation relies upon the quantity of real reckonings included in performing the calculation. The proficiency of the system relies upon that of the calculation and the effectiveness of the code executing the calculation. Notice we incorporated the code for swapping exhibit components inside the circle in choice sort as opposed to calling a capacity to perform this operation. A capacity call obliges included transforming time in place to store contention qualities, exchange project control, and recover the returned worth. At the point when a capacity call is in a circle that may be executed numerous times, the additional transforming time may get to be huge.

TABLE II. SELECTION SORT EXECUTION TIME CONSUMPTION (IN MILLI SECONDS) CHART

Trail No.	No of Values in Dataset							
	1000		3000		5000		7000	
	SC	QC	SC	QC	SC	QC	SC	QC
1	162	154	831	777	1855	1651	2984	2849
2	150	148	853	781	1794	1610	3038	2831
3	173	158	786	748	1795	1667	2957	2823
4	156	149	852	761	1739	1668	3104	2935
5	156	151	787	763	1750	1698	3102	2982
6	154	149	794	765	1805	1644	3047	2837
7	152	147	843	764	1727	1714	2972	2849
8	159	149	784	754	1763	1658	3025	2792
9	160	146	853	799	1772	1661	3033	2728
10	174	152	849	767	1750	1631	3067	2840

SC : Single Core QC: Quad Core

C. Case Study on Linear Search

Straight inquiry or consecutive scan is a strategy for discovering a specific esteem in a rundown that comprises of checking each one of its components, each one in turn and in arrangement, until the fancied one is found.

Straight inquiry is the least difficult hunt calculation; it is an unique instance of animal power look. Its most pessimistic scenario expense is corresponding to the quantity of components in the rundown; thus is its normal expense, if all rundown components are just as liable to be hunt down. Consequently, if the rundown has more than a couple of components, different routines, (for example, parallel inquiry or hashing) may be substantially more effective.A search traverses the entire collection of elements until either the desired element is found or the collection is exhausted. Sequential search is at best O(1) , at worst O(n) , and on average O(n) . If the data being searched are not sorted, then it is a relatively efficient search. However, if the data being searched are sorted, we can do much better[8].

Similarly, Linear Search program is also developed for both single core and quad core and the results are as tabulated in Table III.

TABLE III. LINEAR SEARCH EXECUTION TIME CONSUMPTION (IN MILLI SECONDS) CHART

Trail No.	No of Values in Dataset							
	1000		3000		5000		7000	
	SC	QC	SC	QC	SC	QC	SC	QC
1	2	2	3	2	7	6	7	6
2	2	1	2	2	6	5	12	8
3	3	2	3	2	6	5	7	5
4	2	1	2	1	5	4	7	6
5	2	2	3	3	5	4	10	6
6	2	1	2	2	6	5	7	6
7	2	2	2	1	6	7	7	7
8	2	1	2	2	6	5	9	8
9	2	1	3	2	6	6	8	7
10	2	2	2	1	5	4	8	8

SC : Single Core QC: Quad Core

D. Result Analysis

The mean result analysis table of the three case studies as shown in Table IV, depicts that programs developed on Quad Core produces better performance compared to that of programs developed on Single Core. Though the authors focused on two core areas Sorting and Searching which are widely used in various applications, this parallel processing with multi-core utilization can be extended to other applications that require minimizing the computing time.

TABLE IV. AVERAGE MEAN RESULT ANALYSIS TABLE

Max Values in Input Data	Algorithm → / Processor Type ↓	Bubble Sort (in ms)	Selection Sort (in ms)	Binary Search (in ms)
1000	Single Core	177.9	159.6	2.1
	Quad Core	156	150.3	1.5
3000	Single Core	1248.8	823.2	2.4
	Quad Core	1185.4	767.9	1.8
5000	Single Core	2653.7	1775	5.8
	Quad Core	2561.7	1660.2	5.1
7000	Single Core	4938.9	3032.9	8.2
	Quad Core	4597.2	2846.6	6.7

IV. CONCLUSION

Many applications in computing are time consuming due to the performance measure on huge data and also algorithms operating on this data. In these cases, Parallel processing is implemented which is the ability to carry out multiple operations or tasks simultaneously. If multicore utilization is used with parallel processing then the utilization of computer is increased a lot which benefits users. Software developers are enjoying tremendous improvements in performance if these applications are ported to run on multi-core CPU-based processors by dividing them into logical modules that can be run more efficiently within less time. The present study involving parallel processing can be extended for mobile applications [9] which are rapidly under development for numerous applications.

ACKNOWLEDGMENT

The authors would like to thank Professors of GITAM University, Vizag and Management of Miracle Engineering College, Bhogapuram for their constant encouragement and useful discussions in successful completion present work.

REFERENCES

[1] Burns, C. "Parallel Proto-a prototyping tool for analyzing and validating sequential and parallel processing software requirements", Rapid System Prototyping, 1991. Shortening the Path from Specification to Prototype, Second International Workshop; pp 151 – 160.

[2] Birkinshaw, C.I. ; Croll, P.R. ; Marriott, D.G. ; Nixon, P.A.; "Parallel processing: a safer option for real-time control software", Control, 1994. Control '94. International Conference, pp 916 - 921 vol.2.

[3] Chandio, A.A. Cheng-Zhong Xu ; Tziritas, N. ; Bilal, K., "A Comparative Study of Job Scheduling Strategies in Large-Scale Parallel Computational Systems", Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference, pp 949 – 957.

[4] Yasushi Kiyoki, Takahiro Kurosawa, Kazuhiko Kato, Takashi Masuda ; "The Software Architecture of a Parallel Processing System for Advanced Database Applications", Data Engineering, 1991. Proceedings. Seventh International Conference, pp 220 – 229.

[5] Yunchun Li ; Lianqiang Shan ; Xinxin Qiao; "A Parallel Packet Processing Runtime System on Multi-core Network Processors", Distributed Computing and Applications to Business, Engineering & Science (DCABES), 2012 11th International Symposium, pp 67 – 71

[6] Wang Min; "Analysis on Bubble Sort Algorithm Optimization", Information Technology and Applications (IFITA), 2010 International Forum on (Volume:1); pp 208 – 211.

[7] Wang Min; "Design and analysis on bidirectional selection sort algorithm", Education Technology and Computer (ICETC), 2010 2nd International Conference, pp V4-380 - V4-383.

[8] J. Ian Munro; "On the Competitiveness of Linear Search", 8th Annual European Symposium Saarbrücken, Germany, September 5–8, 2000 Springer Proceedings; pp 338-345.

[9] Sujatha, K; Nageswara Rao, P.V ; Sruthi, K.J.; Arjuna Rao, A; "Design and development of android mobile based bus tracking system", Networks & Soft Computing (ICNSC), 2014 First International Conference; IEEE 2014; pp 231-235.