MULTIPROCESSING OPTIMIZATION - PARALLEL QUICK SORT USING OPEN MP

Himanshu Dewangan\(^1\), Angulakshmi .M*\(^2\) and Nagarajan .I\(^3\)

\(^1\)School of Information Technology and Engineering, VIT University–Vellore,Tamilnadu, India.
\(^2\)Department of Computer Science, SRM University –Chennia, Tamilnadu, India.
\(^3\)Email: angulakshmi.m@vit.ac.in

Received on 26-07-2016 Accepted on 29-08-2016

Abstract

In recent period of Computerization new execution upgrades and a few leaps has been accomplished in range of parallel programming with multi centers frameworks. Current parallel strategies and prerequisite reflects significant execution jumps in projects, however to fulfill parallelism we need to overcome and constitute befuddling, complex and blunder inclined builds. In this article, a parallel system is created utilizing the structure of Open MP API. A successive quicksort program written in C Programming dialect is changed over to parallel fast sort project to accomplish better speed and execution. Changed over system in Open MP build is then checked for different conditions utilizing GCC Open source compiler and run time more than a few size of information and velocity up accomplished is presented in this paper.

Keywords: · Multicore- Parallel processing – Open MP- Quick Sort- Speedup

Introduction

One of the new methodologies used to fabricate the execution of a framework is improving multi center processors. In multi center/core processors, two or more core is used as a piece of solicitation to improve the speed of execution and efficiency of the system. The various advantage of multi core processor is upgrading the execution of activities that have transmission capacity limit, boosting the amount of PC errands that can be performed at the same time, extending users (workers) number where all are utilizing the same server or PC and the improving designing's flexibility to meet new usage models. Nowadays, multi core technology [1] is a flourishing well as PCs having single core processor are accomplishing the uttermost scopes of suitable speed and multifaceted nature. The various organizations like AMD, Mediatek, Intel, Qualcomm, ARM, etc. are developing multicore architecture chips and has good demand in market. Both individual system and embedded system are designed based on Multicore architecture. Flynn's investigative arrangement classifies PC’s as Single-Instruction-Single-Dataflow (SISD), Single-Instruction-
Multiple-Dataflow (SIMD), Multiple-Instruction-Single-Dataflow and (MISD) Multiple-Instruction-Multiple-Dataflow (MIMD). Out of these four classes, MIMD machine can execute different instruction in the meantime in mix of taking a shot at an alternate and free data stream, Hence MIMD model is the most sensible for parallel computing. Open MP is proposed for the synthesis of multi-focus programs. Open MP [2] offers a model that is all around sorted out and clear for framework change, examination and verification of made venture. Any framework that takes a shot at single focus processor will work with multi center processor. Focus of the Open mp is to fulfill parallelism and speedup and to enhance execution.

A venture execution running on a single focus can be redesigned over multi center by part the whole framework into different strings which can all the while continue running on various processors. In view of the growing reputation of multi center headways, various tries of parallelism have been represented in composing. Beside existing techniques, there is adequately still space for making efficient strategy for better parallelism. In this article, multi focus framework streamlining is done where a singular focus sorting calculation [3, 4] (Quick sort) is changed to a parallel programming code fusing Open MP to fulfill speedup and efficient parallelism.

**Open Multiprocessing (OpenMP) Technology**

Open MP (Open multi-processing) is bundle, likewise called Application Programming Interface (API) supporting multiprocessing programming in C, C++, and FORTRAN with shared memory, on most stages, processor structures and working frameworks, including Unix Based, Linux Based, Apple OS X, and Windows working frameworks. Open MP comprises of an arrangement of orders for compiler, environment variables and library schedules that impact runtime conduct.

Multithreaded programming is a strategy for parallelism whereby a parent thread forks a number of slave threads (string) to share the task among them. The strings then run at the same time, with the runtime environment allocating threads to different processors. The segment of code that is intended to keep running in parallel is checked as need, with a pre-processor order that will bring about the threads to shape before the segment is executed. Each thread has an id connected to it which can be acquired utilizing a capacity (calledomp_get_thread_num()) . The string id is a whole number, and the parent thread has an id of 0. After the completion and execution of the parallelized code, the strings join once more into the expert string, which proceeds with forward to the end of the system. As a matter of fact, every string executes the parallelized segment of code freely. Work-sharing builds can be utilized to partition an undertaking
among the strings so that every string executes its designated part of the code. Both assignment parallelism and information parallelism can be accomplished utilizing OpenMP as a part of along these lines.

The runtime environment allots strings to processors contingent upon utilization, machine load and different elements. The runtime environment can allot the quantity of strings taking into account environment variables, or the code can do as such utilizing capacities. The OpenMP capacities are incorporated into a header record named omp.h in C/C++.

**Related Work**

Baderatal [5] in their paper clarified SWARM, an open-source library for essential primitives that completely make utilization of multi-center processors. In [6], they have exhibited a parallelized calculation called Map Sort to quicken Electronic Configuration Mechanization (EDA) virtual products execution on multi-center design. With a specific end goal to accelerate sorting process, multiprocessors are utilized for parallel sorting. Different parallel sorting calculation, for example, bi-tonic sort [7], section sort [8], parallel consolidation sort [8], parallel radix sort [9] and randomized parallel sorting [10] have been created. In [11], a parallel sorting calculation utilizing Representation Preparing Units (GPU's) has been concocted.

Qianan and XuChao examine the parallelism of consecutive calculations of sorting that depend on multi-center frameworks. Nadathur [12] composed superior parallel schedules of union and radix sort for some center GPU's in CUDA. Manetalah [13] built up an efficient parallel calculation psort in C dialect that is perfect with standard qsort.

Any system that utilizations qsort from standard C library can be quickened by just changing qsort call with psort. For nearby successive sorting, psort make utilization of standard qsort as a sub-schedule. So if the qsort execution is enhanced, psort execution will naturally be made strides

**Parallel Quick Sort with Open MP constructs**

Among all existing sorting calculations, quicksort[14] is seen as the most standard and massively used sorting tallies. The quicksort figuring created by Tony Hoary, is a gap and vanquish estimation. In vivacious sort estimation, presentation is conveyed into two by a turn partition or rotate element. Those parts that are more unassuming showed up particularly in association with Turn are moved before it and all sections that are more noticeable than Turn are moved after it.

Moving of parts before or in the wake of swing ought to be conceivable in straight time. The sort estimation is changed over to parallel quicksort utilizing Open MP [15]. We have used GCC Compiler with Open MP innovation. In parallel quicksort, recursive calls are passed on with OpenMP clauses and constructs so they can continue running in parallel.
Conceivable framework to execute the principle cycle of the parallel system is as indicated by the going with steps:

- Select Turn from sub progression and telecast it to each one of the processors.
- Subdivide the information piece available on each processor into two domains using the Turn part.

```c
void qsort_parallel(int l, int r)
{
    if(r-l)
    {
        int pivot=a[(r-l)/2].tmp.
        int less = -1, more = r;
        for(more > less)
        {
            if(a[more] < pivot)
            {
                tmp=a[less1];
                a[less1]=a[more];
                a[more]=tmp;
            }
        }
    }
}
```

Figure 1 Parallel quick sort algorithm using Open MP

The #pragma omp task wait clause shows that the recursive capacity can't be permitted to proceed with its operation before all the solicitations of #pragma omp errand in the same recursive capacity have finished. Our parallel quick sort program: Quick sort principle function is composed with OpenMP Clause is given figure1. To start parallel execution, the setting of Open MP must be made, instated and start execution at an exact section point in the project. In above code, #pragma omp parallel is utilized to make parallel locale, where piece of code composed is executed in parallel and as each project begins as a solitary thread system. #pragma omp single provision is utilized to get it. Figure 2 demonstrates to run the system in GCC Compiler.

Figure 2 Run time system in GCC compiler.

Experimental results

The execution of two calculations can be broke down by considering the real runtime and accelerate Speedup factor. Figure 3 demonstrates the time taken to perform sorting of n Integer number information in two algorithm. It likewise
Angulakshmi M*et al. /International Journal of Pharmacy & Technology exhibit that the parallel quicksort actualized utilizing Open MP perform potentially better over consecutive quicksort, due to the use of parallelism. Between the two sorting algorithm, Parallel quicksort sort the data in less number of time. Figure 4 displays the speedup achieved by Parallel quicksort over Successive quicksort, which is astoundingly poor on account of unbalanced burden adjusting of data.

![Time Take to perform sorting of n integer data](image)

**Figure 3** Time taken to perform sorting of n integer data.

![Speed Up achieved in Parallel Quick sort](image)

**Figure 4** Speed up achieved in parallel quick sort.

**Conclusion**

Advancement of multicore innovation and handling has been moving the key thoughts in the area of programming research exclusively those applications that continues running on PCs and Servers. Activities and applications that are made with considerations of parallel programming not simply improved the execution drastically, it is moreover considered as significant variable for the progression and commonness of multi centre frameworks. Open MP systems demonstrate straight forward and especially composed model for the headway of multi centre and parallel programming. In this paper, two sorting computations are discussed and broke down adequately. The premise correlation is real runtime, number of information feeded and speedup factor accomplished by parallel quick sorting calculation over successive consecutive quick sort. It is watched that parallel sorting performs well in all respects conversely with progressive quicksort. The well performed execution is clear since parallel computations exploits
parallelism to diminish the holding up time. In future, same examination can be analysed with parallel sorting computations (parallel quicksort and hyper quicksort) and parallel sorting by reliable investigating estimation for wide variety of the MIMD models.

References

5. https://sorceforge.net/projects/m multicore-swarm


Corresponding author:
Angulakshmi M*,
Email: angulakshmi.m@vit.ac.in