IJCSC 0973-7391 Volume 7 • Number 1 Sept 2015 - March 2016 pp.59-63 available online at www.csjournalss.com

Performance evaluation of counting words from files using OpenMP

Prof. Smita Agrawal [1,3], Prof. Preeti Kathiria[1,3], Atul Patel[4,5], GUNJAN ASWANI(12MCA02)[2, 3]and BINDIBEN BHATT(12MCA05)[2,3]

[1]Faculty of Comp. Sc. and Eng, Institute of Technology, Nirma University, Ahmedabad – 382481

[2]Student of Comp. Sc. and Engg Department, Institute of Technology, Nirma University, Ahmedabad – 382481

[4]Dean of CMPICA, CHARUSAT University, Changa, Gujarat, India.

[3]{smita.agrawal, preeti.kathiria,,12mca02,12mca05}@nirmauni.ac.in , [5]atulpatel.mca@charusat.ac.in

Abstract: Nowadays every organization has large data and there are many business functions that require the same stored data in some form or the other, which leads to processing of stored data. Faster processing either requires highly efficient CPU with more than one cores or CPU of different machines connected in network. It means for faster processing the data available needs to be distributed on different nodes so that it can be processed independently by every node concurrently and thus it is processed faster. This paper takes one such example of counting words from large number of files in which large number of files can be considered analogous to large data and word count algorithm can be considered as processing task to be performed on files, which then gives word count from each file as extracted information. In this paper two possible methods of processing data are presented 1) Sequential and 2) Parallel along with their performances and factors to be considered. The implementation of both algorithms is shown using OpenMP API in C language with their performance.

General words: Speedup, Performance Improvement, Parallelism, Parallel Processing, Amdahl's law, Communication, Load Balancing

Keywords: Word count from file, Comparison between sequential and parallel execution, word count by different threads, **Open MP**, sequential word counting, Stat system call.

Introduction

Nowadays every organization has large data and there are many business functions that requirethe same stored data in some form or the other, which leads to faster processing of stored data. Processing of data can be done basically in two ways 1) Sequential 2) Parallel processing. Difference between them is as follows:

"In Parallel processing many instructions are carried out simultaneously based on the principle that large problems or data can often be divided into smaller ones, which are then solved concurrently (in parallel).[13]While in serial or sequential programming a problem or data is divided into a discrete series of instructions and those instructions are executed one after another. Only one instruction may execute at any moment in time"[5].Both of the methods have their own pros and cons.

Pros

- Serial/Sequential
    - Simple and easier, good for such problems that can't be parallelized, demands less resources and less expensive [6].

- Parallel
  - Takes less time, better resources utilization, fault tolerance, load balancing.

Cons

- Serial/Sequential
  - Takes more time to complete the problem, no fault tolerance.
- Parallel
  - Needs more resources.
  - As the problem is divided into smaller tasks and these tasks execute concurrently so some mechanism is needed to manage the concurrently executing tasks which may involve some overhead.
  - More expensive than serial or sequential computing.

Every parallel program has two parts: housekeeping and problem processing. Housekeeping involves the coordination of multiple processors, while problem processing is the actual computation. Amdahl's law states that the percentage of time each processor spends on housekeeping increases with the number of parallel processors. The implication of Amdahl's law is that it is impractical to increase the number of parallel processors beyond a certain point".[6]

Applications of parallel processing

"Parallel processing is becoming increasingly important in the world of **database computing**. These days, databases often grow to enormous sizes and are accessed by larger and larger numbers of users. This growth strains the ability of single-processor and single-computer systems to handle the load. More and more organizations are turning to parallel processing technologies to give them the performance, scalability, and reliability they need. **Oracle Corporation** is a leader in providing parallel processing technologies in a wide range of products".[7]

"Weather forecasting provides another real-life example of parallel processing at work. Satellites used for weather forecasting collect millions of bytes of data per second on the condition of earth's atmosphere, formation of clouds, wind intensity and direction, temperature, and so on. This huge amount of data has to be processed by complex algorithms to arrive at a proper forecast. Thousands of iterations of computation may be needed to interpret this environmental data. Parallel computers are used to perform these computations in a timely manner so a weather forecast can be generated early enough for it to be useful". [7][9].

To differentiate between sequential and parallel method this paper shows implementation of one such scenario which is word count from large number of files.

Related work

In the above mentioned applications of parallel processing many factors are there that are to be studied and addressed before opting for any parallel computing models for any specific problem.From them speedup is one of the major and important issue.Speedup is defined as the time it takes a program to execute in serial (with one processor) divided by the time it takes to execute in parallel (with many processors).Speedup is defined as the speed of a program is the time it takes the program to execute. This could be measured in any increment of time.

The formula for speedup is: S=T (1) / T (j), where T (j) is the time it takes to execute the program when using j processors. Efficiency is the speedup divided by the number of processors used. [9]

The second factor to be considered is communication which depends on the nature of the problem. Some problems may need communication and some may not, so accordingly respective model should be adopted and respective programming language must be chosen along with cost, latency and bandwidth of communication. The next one is synchronization which makes sure that all the tasks in the group wait for each other till all of the tasks complete their work. Once all tasks complete the work then the group can proceed. Load balancing is also one of the important factor which tries to equally distribute the available work on existing processors so that none of the processor remains idle that is all of them are involved in processing.

About OpenMP

"OpenMP is based on the shared-memory model. The standard view of parallelism in a shared-memory program is fork/join parallelism. When the program begins execution, only a single thread (master/main thread) is active. The master thread executes the sequential portions of the algorithm. At points where parallel operations are necessary, the master thread forks (creates or awakens) additional threads. Then, the master thread and these new threads work concurrently through the parallel section. At the end of the parallel code, the created threads die or are suspended, and the flow of control returns to the single master thread."[10]

OurApproach

This paper through the case study of counting words from large number of files presents the sequential program of counting task and parallel program of counting task. Along with counting task it also shows the time taken to execute both the programs individually which indirectly indicates the speedup of the programs. For performing the work of counting words from large files both the programs have used the system calls and functions related to file management.

Working Model

This paper shows implementation of counting of words by threads written in C code using OpenMP functions, librariesand pragmas. The algorithm of counting the word implemented in this paper is as follows:

1) Read the directory name.
2) List down the files in specified directory (using Stat call)
3) Iterate through each file from the list and
    3.1 Open the file
    3.2 Read character from file
    3.3 If space or new line is encountered increment the word counter
    3.4 Repeat 3.2 to 3.3 step until end of file is encountered
4) Close the file
5) Repeat 3-4 steps till all entries from the list are visited.

Steps 1, 2 are executed sequentially. From step 3 onwards the block of code is marked as parallel and thus the code is executed by different threads parallel.

Evaluation

Both of the programs were executed on Ubuntu machine that supported 4 cores.In our model list of files in the directory is used as data from which word count is calculated or counted. For performance of parallel code the output of parallel code is compared with our sequential program

of word count. Both are compared based on the time they take to give word count from the file(s) as output. The results were as follows:
The parallel code execution and time details:
Real 0m0.021s
User 0m0.052s
Sys 0m0.000s

Sequential code Execution and time details:
Real 0m0.030s
User 0m0.041s
Sys 0m0.000s

By viewing the results it can be stated that parallel code does word count faster than the sequential program of word count.

Future work

This task can be considered in terms of MAP-REDUCE concept if it is to be implemented using processes.It can be done as follows: Firstly the files are to be distributed on different processes, which can be considered analogous to mapping files on processes.Then each process does its own task of counting the words from file independently.Once the counting is done then each process givesthe count to reducer process which collects counting from all of the processes and displays the count of each file.
This task can be extended further by counting such words from file that has nnumber of occurrence in the file, and dividing the counting on multiple processors so that the counting is done parallel and faster. Logically words in file can be divided into n number of parts and then each part can be assigned to available processes for counting. Other possible solution is as described in above paragraph, large number of files are distributed to different number of processes available and each process may do the counting of words(having n number of occurrence) from the files given to it.

Conclusion

It can be concluded that the above task if done parallel gives you more speedup as compared
if done sequentially, because in sequential there's only one process doing the task one after the other in sequenceand in parallel the work is distributed among several processes (threads) and thus all available processes work independently completing the task faster.
If in parallel the distribution of task takes more time than the time required by the processes to complete the task then in that case parallel will take more time than sequential which is not desirable. So while making the task parallel it should be noted that distribution time is very negligible as compared to time taken by the process to complete the task.One more factor to be kept in mind before applying any parallel technique for existing tasks to be done is that if number of tasks are less than the available number of processors then opting the parallel method for solving tasks may not give the benefit of parallelism as the existing resources (processors won't be utilized in optimized way).
Thus parallelism came into existence because there's a limitation on increasing the processing power of single core (Amdahl's law), hence by introducing multi-core concept parallelism is implemented but it is not easy to incorporate parallelism into solution of existing problem

because to implement it in such a way so that in true sense it improves the efficiency of the problem solution many issues are to be handled or addressed. For example factors such as communication needed or not, load balancing, synchronization, speedupare also to be considered. Also before writing parallel code for the given problem it should be studied and analyzed whether the given problem is parallelizable or not. Once this is decided then only one can proceed to the steps of parallelization.

References :

[1] http://www.slideshare.net/cloveretl/cloveretl-cluster
[2] http://en.wikipedia.org/wiki/Parallel_computing
[3] http://en.wikipedia.org/wiki/Concurrent_computing
[4]http://mivuletech.wordpress.com/2011/01/12/difference-between-sequential-and-parallel-programming/
[5]http://www.akira.ruc.dk/~keld/teaching/IPDC_f10/Slides/pdf/2_OpenMP_Introduction.pdf
[6]http://www.ehow.com/info_8311861_differences-between-serial-parallel-processing.html(Referred in pros and cons)
[7] http://oreilly.com/catalog/oraclepp/chapter/ch01.html
[8] http://en.wikipedia.org/wiki/Amdahl's_law
[9] http://home.wlu.edu/~whaleyt/classes/parallel/topics/amdahl.html
[10] http://msdn.microsoft.com/en-us/library/cc983823.aspx(Open Mp)
[11] Daniel Warneke, Member, IEEE, and Odej Kao,Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud, *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 6, JUNE 2011*
[12] gWord: A tool for genome-wide word search and countXieJianming, Sun Xiao, Lu Zhiyuan, XueWeiyan, Dong Xianjun, Lu ZuhongState Key Laboratory of Bioelectronics, Southeast University, Nanjing, Chinaxiejm@seu.edu.cn, xsun@seu.edu.cn
[13] Das, Swagatam, and PonnuthuraiNagaratnamSuganthan. "Differential evolution: a survey of the state-of-the-art." Evolutionary Computation, IEEE Transactions on 15.1 (2011): 4-31.