Performance Analysis of different Mathematical Algorithms in Parallel Environment

Dilip Kumar Nayak
Student, Raipur Institute of Technology
Chhatouna, India

Mr. Avinash Dhole
Assistant Professor, Raipur Institute of technology
Chhatouna, India

***Abstract*—In Present day, multicore systems have become popular as it provides parallelism and hence less delay, but multicore systems provide only hardware parallelism. In order to achieve best result we should use software parallelism also. To achieve software parallelism there are many programming model like OMP, MPI etc. Now a day's high performance computing mainly center's around parallel computing. Parallel computing is the ability to carry out multiple operations or tasks simultaneously. Ideally, parallel processing makes programs run faster because there are more engines (CPUs or cores) to run the program. Sequence and series like sin series, cos series, arithmetic progression, geometric progression, harmonic progression are most frequently used in mathematical tools. Therefore the parallel computation is an efficient way to improve the performance. By putting some constraints on the data and taking the advantage of the hardware. The performance of the different sequential and serial algorithms can be significantly improved. This paper provides the review of various mathematical algorithms which has developed parallel.**

***Keywords*— Sequence and series, Geometric Progression(g.p.), Harmonic progression(h.p.), Arithmetic Progression (a.p.), Parallelism**

I. INTRODUCTION

The field of numerical analysis predates the invention of modern computers by many centuries .Linear interpolation was already in use more than 2000 years ago. Invention of the computer also influenced the field of numerical analysis, since now longer and more complicated calculation could be done [1].

Geometric Progression is a mathematical tool which is designed for solving a scientific problem more quickly when

- Classic methods are too slow.
- For finding an approximate solution when classic methods fail to find any exact solution [2].

There are various series like: Arithmetic progression, geometric progression, sin series, cos series. A geometric progression is a sequence of numbers where each term after the first is found by multiplying the previous term by a fixed number called the common ratio. The sequence

1, 3, 9, 27 . . .

It is a geometric progression with first term 1 and common ratio 3. The common ratio could be a Fraction and it might be negative.

In general we can write a geometric progression (g.p.) as follows: a, ar, ar^2, ar^3 .the nth term of a g.p. is given by: $\mathbf{ar^{(n-1)}}$

The sum of the first n terms of a g.p. is $Sn=a(1- r^n)/(1-r)$ valid only if $r \neq 1$.The sum of the terms of a geometric progression is known as geometric series [2].

Today's the parallel algorithms are focusing on multi-core systems. The design of parallel algorithm and performance measurement is the major issue on multicore environment. If one wishes to execute a single application faster, then the application must be divided into subtask or threads to deliver desired result.

II. VARIOUS MATHEMATICAL TECHNIQUES

A. Mr D.S. Ruhela and Mr R.N.Jat, "Complexity & Performance Analysis of Parallel Algorithms of Numerical Quadrature Formulas on Multi Core system Using Open MP)".(2014)

The authors studies two version of numerical quadrature algorithms: sequential and parallel. In the experiments the execution times of both the sequential and parallel algorithms have been recorded to measure the performance (speedup) of parallel algorithm against sequential. The result obtained shows a vast difference in time required to execute the parallel algorithm and time taken by sequential algorithm. Based on their study, they concluded that parallelizing serial algorithm using Open MP has increased the performance [1].

B. M. F Mridha, Mohammad Manzurul Islam, Syed Mohammad Oliur Rahman. "A New Approach of Performance Analysis of Certain Graph Algorithms "(2013)

In this paper authors presented a new parallel Prim algorithm that grows multiple trees in parallel. They made simple observations based on the cut property of the graph to grow MSTs in parallel. Their algorithm achieves reasonable speedup when it is compared with Serial Prim algorithm for dense graphs and sparse graphs. The Speedup computed helped realize performance improvements by the use of parallel algorithms. In case of breadth first search algorithm in parallel when graph is sparse speed up is 2.0 while that of when graph is dense 1.9. As for as prim's is concerned speedup is at the minimum of 1.96 i.e. 2.0 more when at least 2 threads are used [4].

C. Mr. Nagraj and Mr. Kumarasvamy,"Serial and Parallel Implementation of Shortest Path Algorithm in Optimization Of Public Transport Travel",(2011)

This paper suggests execution of 3 shortest path algorithms (Dijkstras's algorithm, Bellman Ford algorithm and Ant-

Colony algorithm) serially and parallel (Using OMP).And shows the result that time cost of multithreaded parallel algorithm on dual core system are much faster than the serial algorithm. The parallel running speed can be improved with the increase of number of cores [5].

D. Abdulellah A. Alsaheel, Abdullah H. Alqahtani & Abdulatif M. Alabdulatif "Analysis of parallel boyer moore String search algorithm " ,(2013)

Boyer Moore string matching algorithm is one of the famous algorithms used in string search algorithms. Widely, it is used in sequential form which presents good performance. In this paper a parallel implementation of Boyer Moore algorithm is proposed and evaluated. Theoretically, it proved that the performance of the parallel algorithm is cost optimal with zero overhead. In practical experiment, different sizes of data have been used to show that the parallel algorithm is very faster and better than sequential especially when the data is large.

E. Mr. Sanjay Kumar Sharma and Dr. Kusum Gupta "Performance Analysis of Parallel Algorithms on Multi-core System using OpenMP",(2012

Authors have studied the typical behavior of sequential algorithms and identified the section of operation that can be executed in parallel and presented the execution time of both serial and parallel algorithm for computation of Pi value. They concluded that the parallelizing serial algorithm using Open MP has increased the performance and for multi-core system Open MP provides a lot of performance increase and parallelization can be done with careful small changes. And at last the parallel algorithm is approximately twice faster than the sequential and the speedup is linear [7].

F. Pranav Kulkarni and Sumit Pathare "Performance Analysis of Parallel Algorithm over Sequential using OpenMP"(2014)

The author studied some algorithms like matrix multiplication and Floyd-Warshell Algorithm and found that the algorithms with small data set gives good performance when executed by a sequentially programming. But as data set increases performance of sequential execution falls down where parallel execution is used for large data set then it gives best results than sequential execution. [8].

G. Anupreet Kaur, Pawan Kumar" Performance Analysis of Scheduling Algorithms in Simulated Parallel Environment"

In this paper author had demonstrated the advantages of deploying a scheduling algorithm method in a parallel system. It had presented an scheduling algorithm method and demonstrated its favorable properties, both by theoretical means and by simulations With the proposed scheme, a kind of statistical multiplexing of the incoming traffic over the multiple processors is achieved, thus in effect transforming a network node into a parallel computer. The improvements of processor utilization decrease the total system cost and power consumption, as well as improve fault tolerance.

III. GEOMETRIC PROGRESSION

A geometric progression is a sequence of numbers where each term after the first is found by multiplying the previous term by a fixed number called the common ratio. The sequence

1, 3, 9, 27 . . .

It is a geometric progression with first term 1 and common ratio 3. The common ratio could be a Fraction and it might be negative.

In general we can write a geometric progression as follows:

Geometric progression: a, ar, ar2, ar3 . . .

Where the first term a and the common ratio is r.

Some important results concerning geometric progressions (g .p.) now follow:

The nth term of a g. p. is given by: ar(n-1)

The sum of the first n terms of a g. p. is $Sn= a(1 - r^n)/(1-r)$ if $r \neq 1$

The sum of the terms of a geometric progression is known as a geometric series.

If the common ratio in a geometric series is less than 1 in modulus, (that is $-1 < r < 1$), the sum of an infinite number of terms can be found. This is known as the sum to infinity, S1.

$S1= a/ (1-r)$ provided $-1 < r < 1$

IV. PERFORMENCE METRICS

There are different kind of parameter for evaluating the performance of a system .These performance parameter are the execution time that are evaluate for each thread and cpu utilization.

A. Scalability of parallel algorithms

Increasing number of processor decreases efficiency with fixed problem size. And increasing the amount of computation per processor increases efficiency with fixed machine size. It should possible to keep the efficiency fixed by increasing both the size of the problem and the number of processor simultaneously .Two Scalability metrics are used.

- Number of threads.
- Number of terms

B. Execution Time

Execution time is used to estimate the parallel execution time for each thread to well utilize the processor is in solving the problem.

C. CPU Utilization

CPU Utilization is needed to monitor via the system monitor to determine whether or not it met the criteria intended.

V. PARALLEL ALGORITHM FOR SUM OF GEOMETRIC SERIES

STEP1: Initialize the variables

STEP2: omp_set_num_threads()

// [function that set the maximum thread count at run time.]

STEP3: Set the initial time using omp_get_wtime() function.

STEP4: #pragma omp parallel and #pragma omp reduction

// [syntax of the openmp]

// [create a team of threads that run the code block in parallel] [reduces the code complexity in parallel]

STEP5: #pragma omp sections

// [contains a set of sections and informs that they should execute in parallel]

STEP6: #pragma omp section

// [This informs that the code block that should be executed by a single thread and creation of section depend on the number of threads]

STEP7: #pragma omp critical

// [A block in which only one thread may enter at a time]

STEP8: Assign the initial value to SUM variable

STEP9: Initializes the counter

STEP10: Calculates the TERM and adds with SUM

STEP11: Increment the counter by 1

STEP12: omp_get_thread_num()

// [Runtime function to return a Thread –ID]

STEP13: [END of Parallel Region]

STEP14: Prints the value of SUM

STEP15: STOP

VI. HARDWARE AND SOFTWARE

The Multicore processor specifications used in this work are dual core and quad core and is describe in Table I.

TABLE I. DUAL CORE

Component	Description
# of processor core	2(Dual)
Processor	Intel(R)core™2Duo CPU T6500 @2.10GHz
RAM	2GB RAM
System Type	32bit

TABLE II. QUAD CORE

Component	Description
# of processor core	4(Quad)
Processor	Intel(R)core™i3-3110M CPU @2.40GHz 2.40GHz
RAM	4.00GB RAM
System Type	64 bit

The software required to perform the parallel process are fedora 15 operating system and OMP parallel programming model and system monitor for to check the CPU utilization of the processor for performance measure or Windows visual studio 2010 with OMP support.

The chart and data which is given here are from windows visual studio 2010 with OMP support. Although Program has been implemented and executed in both operating system.

VII. IMPLEMENTATION RESULT

After the execution of parallel program for the different number of threads and for the different number of terms we analyse the average execution time for each thread which is given in following tables(P1=dual core processor,P2=Quad core Processor).

A. The average execution time for each thread in 100 numbers of terms.

TABLE III. ANALYSE DATA FOR 100 TERMS

Number of thread in P1 processor	Average Execution time(in second) in P1	Number of thread in P1 processor	Average Execution time(in second) in P1
Serial(1)	.071471	Serial(1)	0.025675
Parallel(2)	0.072766	Parallel(2)	0.038599
Parallel (4)	0.082612	Parallel (4)	0.031389
Parallel (8)	0.070805	Parallel (8)	0.031218
Parallel (16)	0.095551	Parallel(16)	0.036531
Parallel (32)	0.099105	Parallel(32)	0.048448

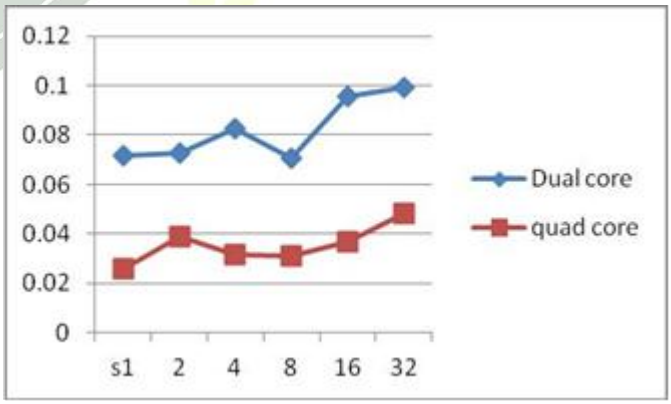


Fig. 1. Average Execution time for 100 terms.

B. The average execution time for each threads in 1000 number of terms

TABLE IV. ANALYZE DATA FOR 1000 TERMS

Number of thread in P1 processor	Average Execution time(in	Number of thread in P1	Average Execution time(in

	second) in P1	processor	second) in P1
Serial(1)	0.272775	Serial(1)	0.142927
Parallel(2)	0.382690	Parallel(2)	0.177757
Parallel (4)	0.354489	Parallel(4)	0.196054
Parallel (8)	0.465510	Parallel(8)	0.183831
Parallel (16)	0.483087	Parallel(16)	0.173315
Parallel (32)	0.484791	Parallel(32)	0.181677

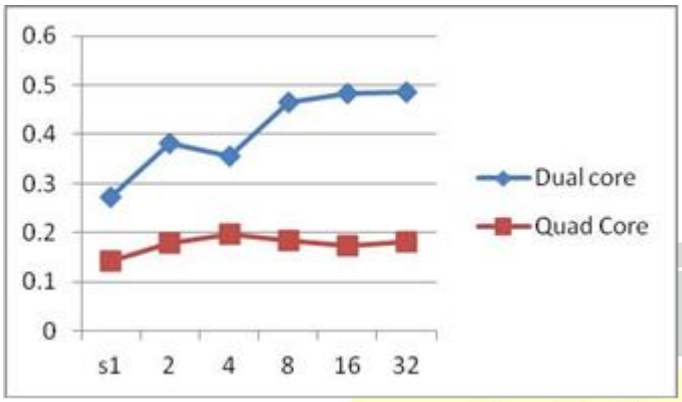


Fig. 2. Average Execution time for 1000 terms.

C. The average execution time for each threads in 10000 number of terms

TABLE V. ANALYZE DATA FOR 10000 TERMS

Number of thread in P1 processor	Average Execution time(in second) in P1	Number of thread in P1 processor	Average Execution time(in second) in P1
Serial(1)	1.206249	Serial(1)	1.039629
Parallel(2)	1.227454	Parallel(2)	1.044320
Parallel (4)	1.222430	Parallel(4)	1.122317
Parallel (8)	1.209286	Parallel(8)	1.151230
Parallel (16)	1.237629	Parallel(16)	1.122573
Parallel (32)	1.228563	Parallel(32)	1.090787

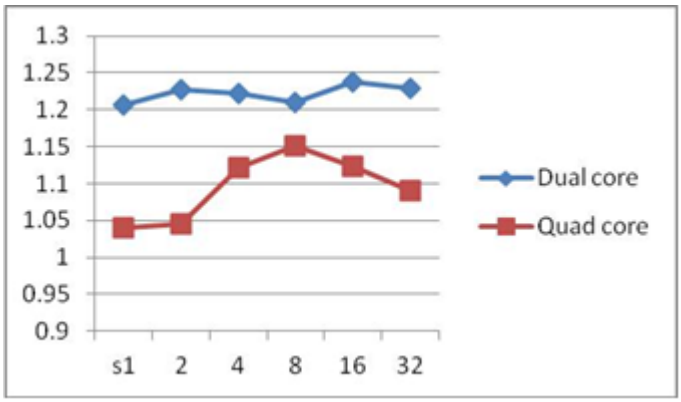


Fig. 3. Average Execution time for 10000 terms.

D. The average execution time for each threads in 100000 number of terms

TABLE VI. ANALYZE DATA FOR 100000 TERMS

Number of thread in P1 processor	Average Execution time(in second) in P1	Number of thread in P1 processor	Average Execution time(in second) in P1
Serial(1)	**10.698792**	**Serial(1)**	**9.383602**
Parallel(2)	**10.726862**	**Parallel(2)**	**9.980090**
Parallel (4)	**10.682143**	**Parallel(4)**	**9.974948**
Parallel (8)	**10.583047**	**Parallel(8)**	**9.077937**
Parallel (16)	**10.598679**	**Parallel(16)**	**8.792485**
Parallel (32)	**10.868552**	**Parallel(32)**	**8.808827**

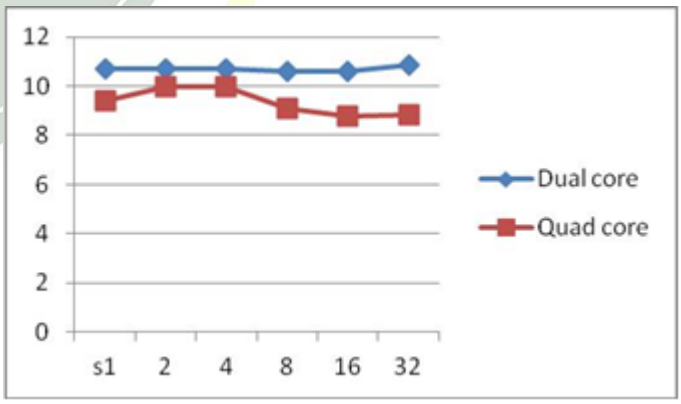


Fig. 4. Average Execution time for 100000 terms.

E. Analyze Average Execution time for each thread for each number of terms.in Quad core

TABLE VII. ANALYZE DATA FOR EACH THREAD ON EACH TERM

Number of term	Average Execution time for each number of thread					
	1	2	4	8	16	32
100	.02565675	0.025930	0.038599	0.031389	0.031218	0.048448

1000	0.142927	0.177757	0.196054	0.183831	0.173315	0.181677
10000	1.039629	1.044320	1.122317	1.151230	1.122573	1.090787
100000	9.383602	9.980090	9.974948	9.077937	8.792485	8.808827

F. Analyze Average Execution time for each thread for each number of terms in Dual core

TABLE VIII. ANALYZE DATA FOR EACH THREAD ON EACH TERM

Number of term	Average Execution time for each number of thread					
	1	2	4	8	16	32
100	.071471	0.072766	0.082612	0.070805	0.095551	0.099105
1000	0.272775	0.382690	0.34989	0.465510	0.483087	0.484791
10000	1.206249	1.227454	1.222430	1.209286	1.237629	1.228563
100000	10.698792	10.726862	10.682143	10.583047	10.598679	10.868552

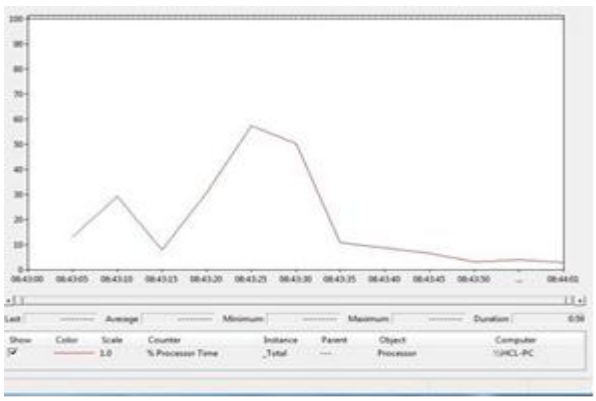
VIII. CPU UTILIZATION

The main objective is to fully utilize the CPU to its utmost potential. Therefore, CPU Utilization needs to be monitored via the system monitor to determine whether or not it met the criteria intended.

CPU Utilization in Dual core processor in percentage of threads.

A. CPU Utilization in 1000000 terms for 2 threads

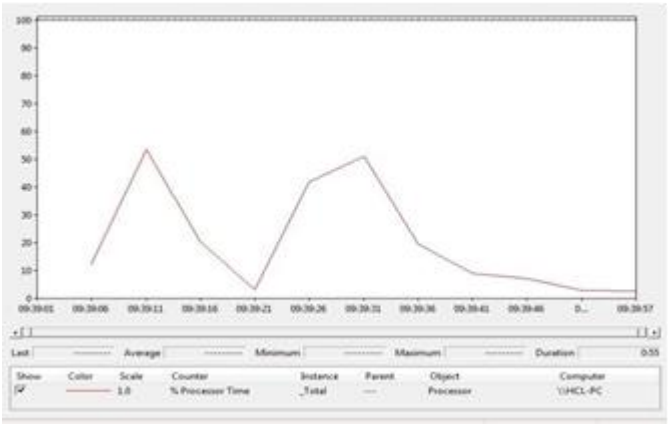
B. CPU Utilization in 1000000 terms for 4 threads



C. CPU Utilization in 1000000 terms for 8 threads

D. CPU Utilization in 1000000 terms for 16 threads

E. CPU Utilization in 1000000 terms for 32 threads



IX. Result Analysis and Discussion

Here we discuss the addition of Geometric Progression series 1+1*2+1*4+…….+1*r^n was making use of 2, 4,8,16 and 32 number of threads for two type of processor first dual core and second quad core. It is observed that in starting when we increase number of thread from 2 to 4 in dual core processor execution time increases, but as we increase number of threads above 4 execution time starts decreasing,. But 32 number of threads time taken is more than the time taken by 2, 4,8 and 16 threads. In quad core processor result is different as we increase number of threads and number of terms execution time increases but a little bit efficiency is noted down from 4 to 8 number of threads. After observation by the graph it is found that.

- 16 is the optimum number of threads for 100, 1000,10000,100000 cycles in dual core processor.
- 8 is the optimum number of threads for 100, 1000,10000,100000 cycles in quad core processor.

In my experiment we divide Geometric Progression Series into different sections, which give data level parallelism. I also used reduction method for code optimization. Execution of Geometric Progression Series on different number of threads gives thread level parallelism. Best condition is when number of threads are in power of 8. It is also observed that if it is increase number of thread beyond number of core (for example 4) then it first complete 4 threads then start the execution for other threads. CPU utilization is best when we use number of threads 32.

Conclusion

In this work effect of parallelization on execution time was studied, addition of Geometric progression was done, by using threads. Thread is an independent smallest unit of processing that can be scheduled by operating system. The problem was divided into increasing number of threads. It was found that upon increasing the number of threads (parallelization) time is increasing. This is probability because of small configuration of microcomputers which include desktop; laptop another reason may be use of OpenMP which has very limited facility of parallelization. Small time also spends in overhead communication. It may be possible that on mini and mainframe computer along parallel operating system and parallel compiler upon increasing parallelization time may reduce. We conclude that for best performance when number of threads are in power of 8.

If any programs are running on quad core processor without thread then it is as good as running a program on single core computer. To make use of quad core computer he should used 4 threads in suitable environment like OpenMP or any other equivalent.

Future Work

The future work can be computation may be made for other series also. This may be extended to binomial series, sequential series integration series another type of complex series like sine series, cosine series etc.

References

[1] D.S. Ruhela and R.N.Jat .” Complexity & Performance Analysis of Parallel Algorithms of Numerical Quadrature Formulas on Multi Core system Using Open MP” Volume 3 Issue 7 July, 2014 International Journal Of Engineering And Computer Science.

[2] Sequences And Series [Online] http://Ltcconline.Net/Greenl/Courses/103b/Seqseries/Seqser.Htmanization” Tmh.As Retrieve On Date 10/01/15.

[3] [Online] http://Www.Wyzant.Com/Help/Math/Precalculus/Series_And_ Sequences. As Retive On Date 10/01/15.

[4] M. F Mridha, Mohammad Manzurul Islam, Syed Mohammad Oliur Rahman. “ A New Approach of Performance Analysis of Certain Graph Algorithms “International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 9, September 2013.

[5] Mr. Nagraj and Mr. Kumarasvamy,”Serial and Parallel Implementation of Shortest Path Algorithm in Optimization Of Public Transport Travel”,international jounal of computer science engineering and information technoo.

[6] Abdulellah A. Alsaheel, Abdullah H. Alqahtani & Abdulatif M. Alabdulatif “Analysis of parallel boyer moore String search algorithm “Global Journal of Computer Science and Technology Hardware & Computation Volume 13 Issue 1 Version 1.0 Year 2013.

[7] Mr. Sanjay Kumar Sharma and Dr. Kusum Gupta “Performance Analysis of Parallel Algorithms on Multi-core System using OpenMP”, International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol.2, No.5, October 2012.

[8] Pranav Kulkarni and Sumit Pathare “Performance Analysis of Parallel Algorithm over Sequential using OpenMP” IOSR Journal of Computer Engineering (IOSR-JCE) Volume 16, Issue 2, Ver. X (Mar-Apr. 2014)

[9] Pranav Kulkarni, Sumit Pathare “Performance Analysis of Parallel Algorithm over Sequential using OpenMP” IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. X (Mar-Apr. 2014).