IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

Performance Evaluation of Duty Cycle Modulation in HPC

Aparna Sure[1], Dr S.L.Deshpande[2]

Student, Computer networks Engg, VTU PG Centre, Belagavi, India [1]

HOD, Computer networks Engg, VTU PG Centre, Belagavi, India [2]

Abstract: High Performance computing (HPC) is the use of parallel processing for running application program efficiently, reliably and quickly. HPC is also refers to how fast one can get the result, and how efficiently can get the result. HPC can also refer as high productivity of systems. Speedup of high performance is measured by considering the parameters like threads. Here 3 cases are considered that is; keeping the number of jobs constant and varying number of threads, keeping number of threads constant and varying number of jobs. And modifying the duty cycle of the threads. The numbers of runs are considered and founded the ideal number of threads to gain the average speedup.

Keywords: Number of threads, number of jobs, average speedup.

I. INTRODUCTION

High Performance Computing (HPC) is achieved by aggregating computing power of different nodes, cloud nodes, or end system, so that today's request demands can be fulfilled with faster response. In today's era HPC is running in almost all or all fields. Like scientific application, medical application, weather forecasting, satellite application, domestic application, industries organization and banking, myriad defense and aerospace applications. HPC currently is playing a role in: training and simulation; on-board systems for navigation, defense, and attack; and command, control, communications, intelligence, computers, surveillance, and reconnaissance.

The performance of HPC is measured in FLOPS -Floating point operation per second. The importance of HPC is in Simulate a bio-molecular of 10000atoms, Non-bond energy term that is 10^8 operations, for 1 microsecond simulation that is 10^9 steps and 10^{17} operations, need to do large number of simulations for even large molecules, for the biggest calculation in servers, for Data Centers.

Applied application of HPC is in Blood Flow in human vascular network, Earthquake simulation, Homogeneous Turbulence. HPC System must have good combination of Multistage (pipeline) functional unit, multiple central processing unit, multiple cores, and operating system. And these systems must perform fast central register, Fast memory transaction, very fast communication among functional unit, and software that integrate these function.

Computation is very important in High performance computation. Maximum speed is achieved in terms of both hardware and software. HPC deals the complex and large problem by dividing and conquer method. Today's higher computation is achieving systems are multicore architecture, parallel, and pipelined systems.

The following parameters are considered to gain the higher performance; the type of algorithm used, number of stages in pipeline, parallelism, concurrency, number of threads. CPU executes instructions faster by employing pre fetching and pipelining. That is fetching the instruction before it executes. That processor fetches reads and decodes an instruction while another instruction is executing.

So to increase the speed of execution CPU Design can be

1. Reduced instruction set computer (RISC).
2. Multiple core processors.
3. Vector processors.

Possible methods to achieve the HPC

Higher computation power can be achieved by parallel architectures

1. Single Instruction Single Data (SISD) Sequential machines.
2. Multiple Instructions Single Data (MISD).
3. Single Instruction Multiple Data (SIMD).

Parallel Problem solving methodologies

1. Data parallelism.
2. Pipelining.
3. Functional parallelism.

Performance Issues:

1. Speed up = Time for sequential code / Time for parallel code

$Sp = Ts / Tp \qquad 1 <= Sp <= P$

2. Efficiency

$Ep = Sp / P \qquad 0 < Ep < 1$

$Ep = 1 > = Sp = P \qquad 100\%$ efficiency

The strongest argument is Amdahal's Law

$S = 1 / f = (1-f) / p$

Where 'f' is the sequential part of the code.

II. PROPOSED WORK

In This thesis performance evaluation of duty cycle modulation is done, and speedup is calculated. As considered there is number of jobs to be executed and that are executed parallel by number of threads. As threads will execute single task fast, and threads can be executed parallel so that jobs can be executed fast with in a shorter

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

period of time. Here the results are measuring and modifying the duty cycle. Considered things are:
1. Number of jobs varying from 8189 to 1000.
2. Number of Threads varying from 5 to 20.

There are 3 cases are considered to measure the results:
1. keeping the number of threads constant and varying the numberofjobs.
2. Keeping the number of jobs constant and varying the numberofthreads.
3. Adjusting the duty cycle of fast executing threads.

Case1:
keeping the number of threads constant, considered constant threads are 5. Varying the jobs that are from 8189, 6000, 4000, 2000, and 1000. There are 5 types of constant jobs are taken to execute.
For 1 type: five threads, 8189 jobs to execute. The thread pool of 5 threads which are executing paralley. Threads named from T1 to T5.

Any one of these threads can execute first there is no such sequence is too followed. They will pick any one of the job out of 8189 jobs .these threads are synchronized , and mutual exclusion is achieved . So that no 2 threads rush to execute the same job at the same time and never lead to the dead lock state environment.

There are some threads which will executing very fast so that they will going to execute more number of jobs as compared to thread which is executing slow. The individual thread execution time is calculated and numbers of jobs that particular threads executing. And overall execution time of job is also calculated. It will run for several times and all threads execution time, number of jobs executed and overall execution time is listed for 20 runs in a table format.

Parallel execution time, speedup is calculated .and correlation coefficient of parallel execution time and serial execution time is calculated. Like the way case 2 is also done and results are analyzed in case 3.that is by observing and measuring the results. Fast execution thread can be found, so that duty cycle of that thread is adjusted.

That is in this thesis it will come to know that what are the efficient number of threads are required to execute. That run time threads can be reduced, and finding the ideal number of threads even though more number of jobs to be executed. Usually if there is more number of jobs for execution then more number of threads is kept for execution so that higher performance is gained.

But it will be expensive in terms of memory, computation cost and energy level of system. So to minimize this issue, thesis helps to find the ideal state of system and ideal number of threads for execution. The system configuration is dual core processor, 2 GB of RAM, fedora 26 operating system, and eclipse software used for programming. Java language is used for coding.

III.RESULT ANALYSIS

In First case: considering 5 threads and varying number of jobs. The below tables show the result analysis.
Table 1: Thread Execution time, parallel execution time, speed up , Average speed up and correlation coefficient.

	T1	T2	T3	T4	T5	Total Execution Time	Parallel Execution Time	Speed Up	Average Speed up	correlation coefficient
1	715	727	665	700	687	4385	3494	1.2550	1.3365	0.9975
2	647	616	607	614	601	3845	3085	1.2464		
3	653	645	598	579	619	4169	3094	1.3474		
4	408	396	401	380	400	3022	1985	1.5224		
5	523	518	516	481	442	3687	2480	1.4867		
6	679	646	664	676	620	4099	3285	1.2478		
7	592	622	645	563	575	3772	2997	1.2586		
8	539	547	568	533	534	3387	2721	1.2448		
9	748	799	782	805	764	5040	3898	1.2930		
10	540	547	518	568	522	3523	2695	1.3072		
11	619	659	626	659	660	4475	3223	1.3885		
12	830	844	862	826	820	5208	4182	1.2453		
13	646	639	633	635	624	4092	3177	1.2880		
14	508	504	504	522	487	3659	2525	1.4491		
15	408	394	396	394	392	2930	1984	1.4768		
16	331	323	346	323	296	2528	1619	1.5615		
17	633	633	647	601	630	4088	3144	1.3003		
18	612	627	618	617	614	3978	3088	1.2882		
19	777	786	798	883	832	5096	4076	1.2502		
20	2703	2679	2724	2635	2709	17131	13450	1.2737		

Table 2: optioned correlation coefficient.

Number of jobs	correlation coefficient
8189	0.9975
6000	0.9881
4000	0.961
2000	0.9671
1000	0.9953

Table 3: Average speed up

Number of jobs	Average Speedup
8189	1.3365
6000	1.286
4000	1.2992
2000	1.2875
1000	1.2757

Case II:
Varying number of threads and keeping constant number of jobs.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

Table 4: Thread Execution time, parallel execution time, speed up , Average speed up and correlation coefficient.

	T1	T2	T3	T4	T5	Total Execution Time	Parallel Execution Time	Speed Up	Average Speed up	correlation coefficient
1	533	536	535	520	500	3322	2624	1.2660	1.2841	0.9975
2	1591	1579	1639	1610	1547	10728	7966	1.3467		
3	633	600	656	593	587	4014	3069	1.3079		
4	549	529	528	524	474	3275	2604	1.2577		
5	564	501	552	489	551	3332	2657	1.2540		
6	643	639	617	621	647	4013	3167	1.2671		
7	493	497	523	529	514	3273	2556	1.2805		
8	612	573	627	635	635	3836	3082	1.2446		
9	526	545	528	540	482	3370	2621	1.2858		
10	567	567	587	602	575	3616	2898	1.2478		
11	499	499	504	558	483	3223	2543	1.2674		
12	374	360	377	346	355	2513	1812	1.3869		
13	444	400	416	434	395	2746	2089	1.3145		
14	537	500	514	496	523	3203	2570	1.2463		
15	506	458	517	501	484	3075	2466	1.2470		
16	601	602	544	567	613	3680	2927	1.2573		
17	510	527	519	478	515	3293	2549	1.2919		
18	366	379	350	375	372	2587	1842	1.4045		
19	622	628	654	654	619	3955	3177	1.2449		
20	597	563	550	540	569	3564	2819	1.2643		

Table 5: Optioned correlation coefficient

	correlation coefficient				
Jobs	5 Threads	8 Threads	10 Threads	15 Threads	20 Threads
8000	0.9975	0.9911	0.9927	0.9731	0.9969

Table 6: Average speed up

	Average Speedup				
Jobs	5 Threads	8 Threads	10 Threads	15 Threads	20 Threads
8000	1.2841	1.2078	1.1679	1.2095	1.1648

As observed the average speedup is increased as the number of jobs increased and with proper number of thread.

REFERENCES

[1] SriduttBhalachandra "Using Dynamic Duty Cycle Modulation to improve energy efficiency in High Performance Computing".
[2] Allan K. Proterfield "Power Measurement and Concurrency Throttling For Energy Reduction in OpenMp Programs".
[3] Peter H. Mills "Software Issues In High Performance Computing and a framework for the development of HPC application.
[4] Abhishek Gupta "Towards Efficient Mapping, Scheduling, and Execution of HPC Applications on Platforms in Cloud "
[5] SébastienVarrette"HPC Performance and Energy-Efficiency of the OpenStack Cloud Middleware".
[6] Daniel Chavarría-Miranda "High-performance computing (HPC): Application & use in the power grid".
[7] Waseem Ahmed"Introducing high performance computing (HPC) concepts in institutions with an absence of HPC culture".