ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Benchmarking of HPC Application on Many Core Architecture

Nisha Bonde, Prof.Y.B.Gurav

M.E Student, Dept. of Computer, Padmabhooshan VasantdadaPatil Institute of Technology, Pune, India

Assistant Professor, Dept. of Computer, Padmabhooshan VasantdadaPatil Institute of Technology, Pune, India

ABSTRACT: The future of computing is parallel. High Performance Computing (HPC) commonly referred to as supercomputing is being used continually to enhance a quality of life. Due the rapid advancements and technology refresh in computer architecture, it is becoming increasingly difficult to gauge the real life performance of large computer systems based on specifications. Performance comparison of multicore and many core processors (Intel Xeon Phi coprocessor) are done by using the well know benchmark suites. Computer systems are designed by manufacturers keeping in mind the full utilization of the hardware by the users. HPC programmers are mostly interested in: (1) Performance – on all the systems they work on (including future ones too). (2) Portability – no need to rewrite just because a new architecture comes along. (3) Maintainability – code should be written in such way that, it can be changed without massive efforts, preferably by the scientists or researcher who are directly uses the applications [8]. There are many more scientific domain application are running on many core architecture. This paper discusses the architecture overview of many core system and Monte Carlo method which is running on such a hybrid system.

KEYWORDS: High Performance Computing, Intel Xeon Phi Coprocessor, Many Core, Multicore, Performance, Portability**.**

I. INTRODUCTION

Most of the computer architectures make robust demands than ever on writers of scientific code. The cost of moving data is increasing by the day as compared to the cost of computing. Earlier, High Performance Computing essentially focused on a single box large scale system. But the design of such mammoth systems will lead to a development of architectures that are different and extremely exclusive from other contemporary architectures. HPC is a tool which not only solves all the grand challenge problems but is now the essence of all possible services available to human beings [1]. The demand for computing is increasing and in particular, the scientific domain is the largest consumer of compute cycles to carry out applied research. HPC is pooling of a large number of computing resources together to run applications in parallel. The Intel Xeon Phi Coprocessor family provides outstanding parallel performance. It is an excellent choice for compute intensive workloads, such as Monte Carlo, HPL, Life Science, Black-Scholes, and many other scientific applications. Monte Carlo methods are a broad class of computational algorithms [6]. The proposed work is based on the Monte Carlo methods; running on normal Intel Xeon Processor as well as on Intel Xeon Phi Coprocessor and comparison of the execution time on both the architecture. Additionally, optimization and vectorization on the code is carried out to improve the performance of the application.

II. RELATED WORK

This paper covers the latest hardware of multicore and many-core architecture as well as the scientific application running on such hybrid architecture. The algorithm of an application can be changed, such that it will utilize the advantages of available hardware architecture. It will help the scientific community to improve the performance of their scientific code and concentrate on their domain-specific work. There are different ways to run the application on such hybrid system. It also maintains the portability of application for future systems

A. *Sandy Bridge Architecture*

Intel has developed micro architecture like Westmere, Nehalem, Sandy Bridge, Ivy Bridge, Haswell, Broadwell etc. Sandy bride is a code name for the micro architecture which is replaced by Nehalem processor. Intel demonstrated a

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Sandy Bridge processor in 2009, and released first products based on the architecture in January 2011 under the Core brand [2]. As shown in Fig 1, the Sandy Bridge-based node has two Xeon E5-2670 processors, each with eight cores. The frequency of each processor is 2.6 GHz, with a peak performance of 166.4 Gflop/s. The total peak performance of the node is therefore 332.8 Gflop/s. Each core has 1.5K-ops, 64 KB of L1 cache (32 KB data and 32 KB instruction) and 256 KB of L2 cache. Last level cache (LLC) of 20 MB has shared by all eight cores called L3 cache. The on-chip memory controller is there which supports four DDR3 channels running at 1600 MHz, with a peak-memory bandwidth per processor of 51.2 GB/s (and twice that per node)[2].Two QPI links are used by each processor to connect with the second processor of a node to established a non-uniform-memory access (NUMA) architecture [2]

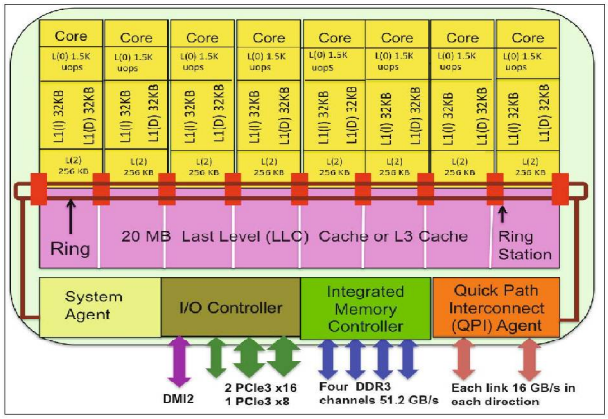

Fig 1 Architecture of Sandy Bridge Processor

B. ***Intel Xeon Phi Architecture***

The Intel Xeon Phi coprocessor is very useful for applications which are highly parallel and threaded. To run the application on Xeon Phi cards, new programming model or language is not required. Only with few changes in the available programming language, we can execute the code on such architecture. This can be done by using an array of general purpose cores with multiple thread, caches, wide vector units, and high bandwidth on die and memory interconnect [7]. Knights Corner is 22nm technology and based on x86 architecture. It is the first product in the Many Integrated Core (MIC) family from Intel. The aim of developing such product is to facilitate the Exascale era of computing. The Intel Xeon Phi coprocessor consists of 60 or 61 in-order cores. Each core consists of 4 threads to hide memory and instruction latency [1]. It vector units are 512 bits wide and execute 8-wide double-precision or 16-wide single-precision single-instruction multiple-data (SIMD) instructions in a single clock[3]. The KNC is running at 1 GHz frequency and delivers up to around 1 or 2 Tflop/s floating-point performances in double- and single-precision, respectively. The KNC has two levels of cache: L1 caches which is 32 kB and a larger globally coherent second level cache (L2) that is divided among the cores. Each core has a 512 kB partition [1]. These cards come in a variety of configurations to address diverse hardware, software, performance, workload, and efficiency requirements [6]. E.g. 7100 series: It provides the turbo boost technology, high memory capacity. 5100 series: Optimized for memory bandwidth workload and memory capacity workloads. 3100 series: Provides performance for compute intensive workloads. We can run applications on these cards using different modes, like native, offload, symmetric and automatic offload. Fig 2 shows the architecture overview of an Intel Xeon Phi card.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

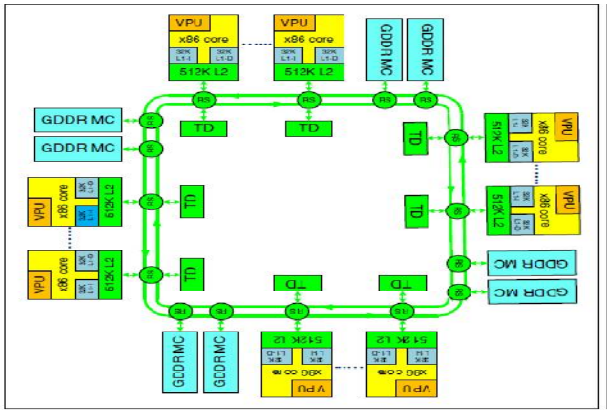

Fig 2 Intel Xeon Phi Coprocessor Architecture

C. ***Monte Carlo Methods***

Monte Carlo methods are one of the broad class of computational algorithms that depend on repeated random sampling to obtain numerical results. These methods are often used in physical and mathematical problems. These methods are useful when it is difficult or impossible to use other mathematical methods. Monte Carlo methods are mainly used in optimization, numerical integration, and probability distribution [5]. Monte Carlo simulation methods do not always require truly random numbers to be useful. Deterministic, pseudorandom sequences techniques are used to test and re-run the simulations. The application should pass a series of statistical tests. Testing whether the numbers are uniformly distributed or follow another preferred distribution, when a large sufficient number of elements of the sequence are considered. A weak correlation between successive samples is also often desirable/necessary [5]. Pseudo-random number sampling algorithms are used to transform uniformly distributed pseudo-random numbers into numbers that are spread according to a given probability distribution [5]

III. PROPOSED METHODOLOGY AND DESIGN

A. ***Ways to compile the application on Xeon Phi cards***

Native compilation: In this mode, the application is compiled using –mmic flags which enables the compiler to generate an object file for Xeon Phi coprocessor [4]. We can run the application from host system by setting some environment variable. Also we can run the application by coping it on the card.

Symmetric compilation: In this mode, the application is run on both the Xeon processors and the Xeon Phi coprocessor. We have to create the executable for both the processor and coprocessor. The MPI processes and related MPI Communication can be done on host CPU and coprocessor. Accordingly all environment variables are set.

Offload compilation: In this mode, user can offload the compute intensive part which is highly parallel by using some Open MP-like pragmas on to the Xeon Phi card. Whenever the Intel compiler finds the offload pragma, it generates the code for both processor and coprocessor. The compiler automatically transfers the code to the coprocessor and there are various clauses which we can used to copy the variable, function to the coprocessor and back to the processor.

B. ***Problem Definition***

Intel Xeon Phi coprocessors are designed to minimize the power required to run any application on the hybrid system. We can fully utilize the scaling and vector capabilities of both processor and coprocessor. For such applications, we can further optimize the application on Intel Xeon Phi coprocessor and also on Intel Xeon processors and address the scalability and efficiency issue by comparing the result on both coprocessor and processors with help of Monte Carlo Methods

C. ***System Design***

The correlation functions are calculated using Monte-Carlo simulations [5]. Fig 3 shows the architecture of the proposed system which includes Intel Xeon Phi coprocessor(s) as well as Intel Xeon processor. The application will run

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

on both the system and further improve the performance with some optimization and vectorization techniques. Intel Xeon Phi is very power efficient coprocessor.
The aim of the propose system is

- Study the latest accelerated architectures such Intel Xeon Phi coprocessor.
- Identify a widely used code using the Monte Carlo method algorithm.
- Port the code onto a Processor based machine, execute the code, and observe the performance with test case datasets.
- Port the same code onto Intel Xeon Phi Coprocessor, execute in native and offload modes, observe performance.
- Optimize the runs onto the Intel Xeon Phi to extract best possible vectorization performance.
- Compare and tabulate the performance figures on the processor, accelerator as well as the combined architecture.

We chose to apply Monte Carlo methods because of their versatility. The basic strategy of Monte Carlo methods is to reduce the variance by averaging the results of large numbers of samples

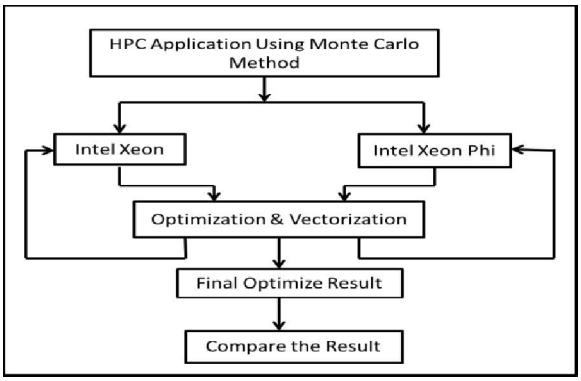

Fig 3 System Architecture of Proposed system

D. ***Implementation Plan***

We are proposing to implement the project to test the scientific application on hybrid architecture which includes many core as well as multicore processor. The detailed plan is explained below.

Module 1: Framework building, Identify a widely used code that uses the Monte-Carlo simulations method. Monte Carlo simulation is widely used in many scientific applications for random number generation, numerical distribution etc.

Module 2: Porting and executing a application on normal Intel Xeon Processor as well as on Intel Xeon Phi Coprocessor using the different running programming methodology like native, symmetric or offload modes.

Module 3: Optimization and vectorization of the code to improve the performance. The objective of this framework is to achieve the highest performance with the best tools and library available along with the hardware capabilities.

Module 4: Result Analysis and Testing. The final module will display the execution time on both Intel Xeon and Intel Xeon Phi Coprocessor and compare the results for analysis purpose and performance of hybrid architecture.

E. ***Mathematical Model***

To calculate the value an option, Monte Carlo simulation uses the risk-neutral valuation method. Let's consider the current price C of a stock option for a particular stock and provides a payoff at time T. Assuming the interest is constant, derivatives value can be calculated as follow [9]:

1. Sample a random path for C in a risk-neutral world.
2. Payoff can be calculated from the derivative.
3. Repeat step 1 and 2 to get many sample values of the payoff from the derivative in risk-neutral world.
4. The mean of the sample payoffs is calculated, to get an estimate of the expected payoff to minimize the risk.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

5. Discount the expected payoff at the risk-free rate to get an estimate of the value of the derivative

Consider the equation uses by the market variable to minimize the risk as:

$$dC = \mu C dt + \sigma C dWt \quad (1)$$

Where dWt is a Wiener process, μ is the expected return in case of minimum risk, and σ is the volatility.

To simulate the path followed by lnC, the period of derivative is divided into N short intervals of length t and we get the equation (2) as:

$$\ln C(t + \Delta t) - \ln C(t) = \left(\mu - \frac{\sigma^2}{2}\right)\Delta t + \sigma\epsilon\sqrt{\Delta t} (2)$$

At the time of expiration the values of European option can be calculated by using call and put options.

$$f_{call}(C_T, T) = \max(C_T - X, 0.0) \quad (3)$$

$$f_{put}(C_T, T) = \max(X - C_T, 0.0) \quad (4)$$

Using Monte Carlo, we can generate N numeric samples with the required (0, 1) distribution, considering the Wiener process, then averages the possible end period stock profits, corresponding to each of the sample values:

$$f_{call}(C_T, T) = \frac{1}{N+1}\sum_{k=0}^{N} \max(C_{k\Delta t} - X, 0) \quad (5)$$

$$f_{put}(C_T, T) = \frac{1}{N+1}\sum_{k=0}^{N} \max(X - C_{k\Delta t}, 0) \quad (6)$$

The result is still the future value of the option at time T. Discounting this value by a factor of e^{-rT}, we get the present value of European call options.The standard error for Monte Carlo converges at the rate of$o\left(\frac{1}{\sqrt{N}}\right)$. The advantage of Monte Carlo simulation is that it can be used when the payoff depends on the path followed by the essential variable C leading toward the expiration and the situation when payoffs take place multiple times during the life of the option. It is particularly useful when the payoff function involves multiple independent variables

IV. RESULTS ANALYSIS

Monte Carlo European Option Pricing code can be run on the Intel Xeon Processor and Intel Xeon Phi coprocessor in single precision and double precision mode. There are various test cases can be performed.

1) Test case on Intel Xeon Processor: In this case the code is compiled with different compiler option like – no-vec, O3, O2 etc. The code is compiled in single precision (SP) and double precision (DP) mode. Fig 4 shows the speedup ratio and efficiency of the Monte Carlo European Option Pricing code compiled in SP mode on Intel Xeon Processor.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

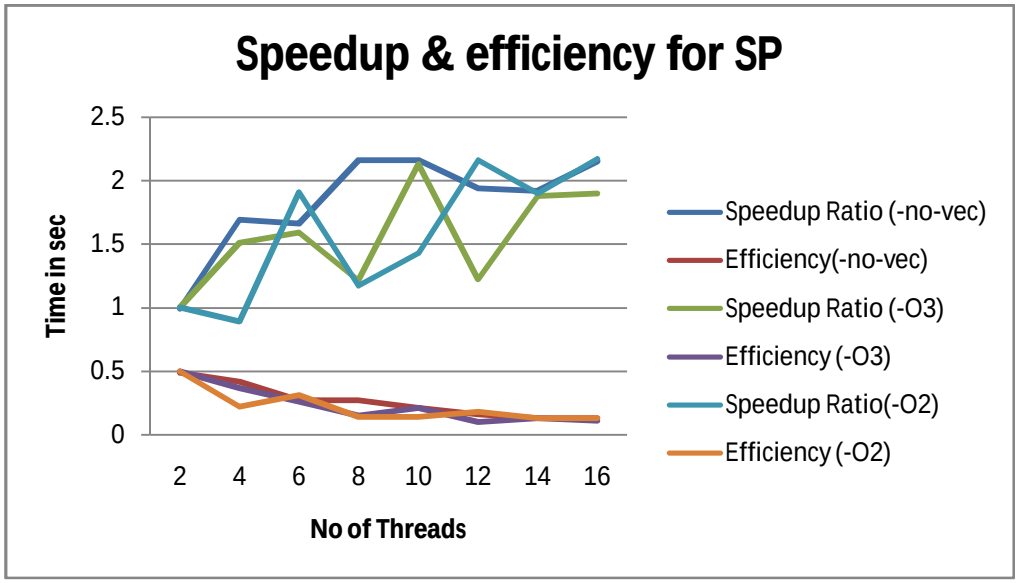

Fig 4. Result on Intel Xeon for Single Precision

From the Fig 4, the efficiency and speedup with –no-vec option, we can see that as we increase the number of threads the speedup increases and efficiency decreases, but in this case the load (i.e the number of options) is increased. Fig 5 shows the speedup ratio and efficiency of the Monte Carlo European Option Pricing code compiler in DP mode on Intel Xeon processor

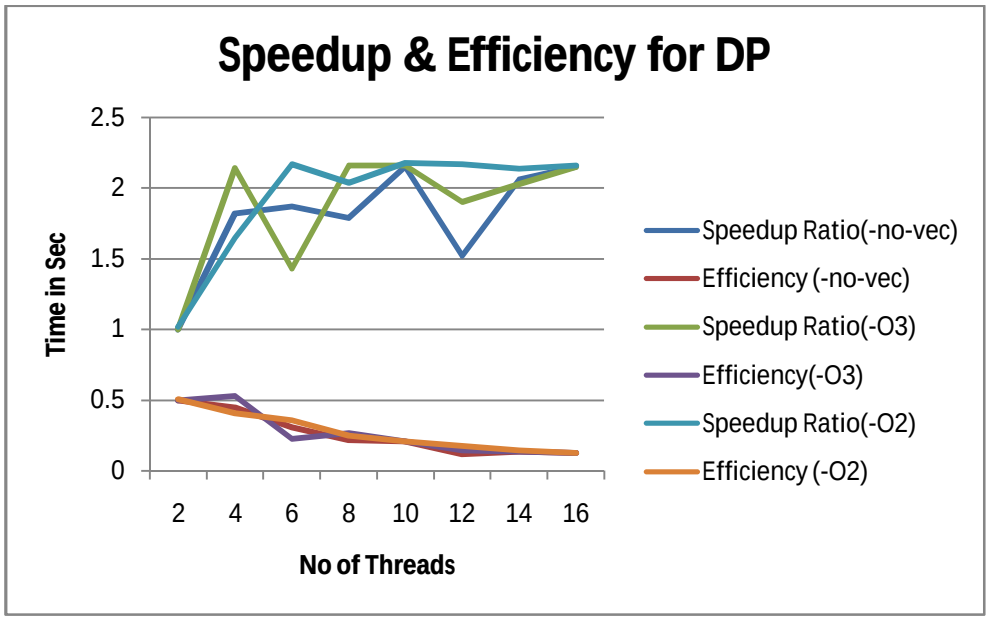

Fig 5. Result on Intel Xeon for Double Precision

From Fig 5, the efficiency for no-vec, O3, O2 is decreaed as we increase the number of threads. The speedup ratio for no-vec option is linear, but for O3 and O2, it is gradually increases and gradually decreases.

2) Test case on Intel Xeon Phi Coprocessor: In this case, the code is running on Intel Xeon Phi coprocessor with different number of threads and different thread affinity variable.Fig 6 shows the result of running Monte Carlo

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

European Option Pricing code in SP mode on Xeon phi card. For affinity variable 'balanced' the code is scaling linearly.

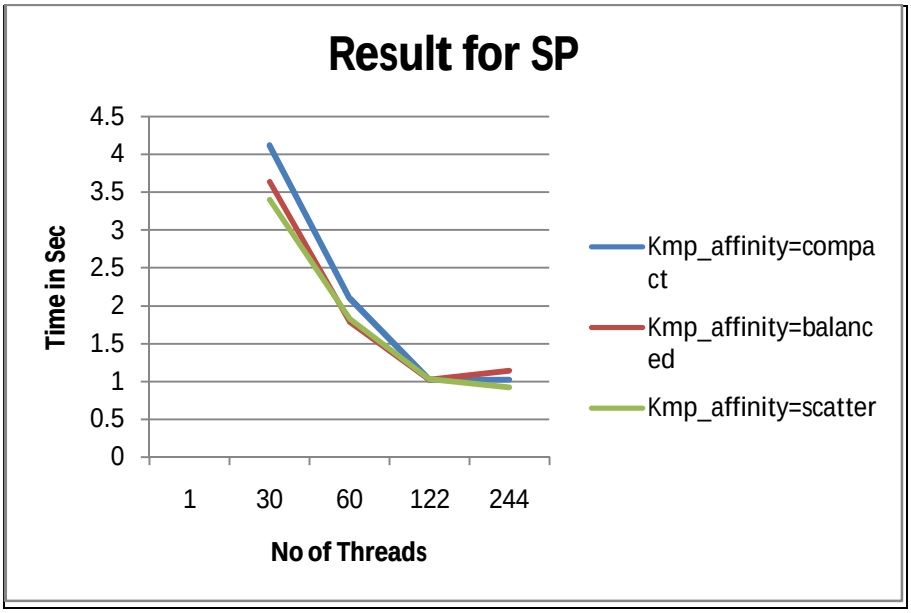

Fig 6 Result for Single Precision on Intel Xeon Phi

Fig 7 shows the result of running Monte Carlo European Option Pricing code in DP mode on Xeon phi card. For affinity variable 'balanced' the code is linearly scaling.

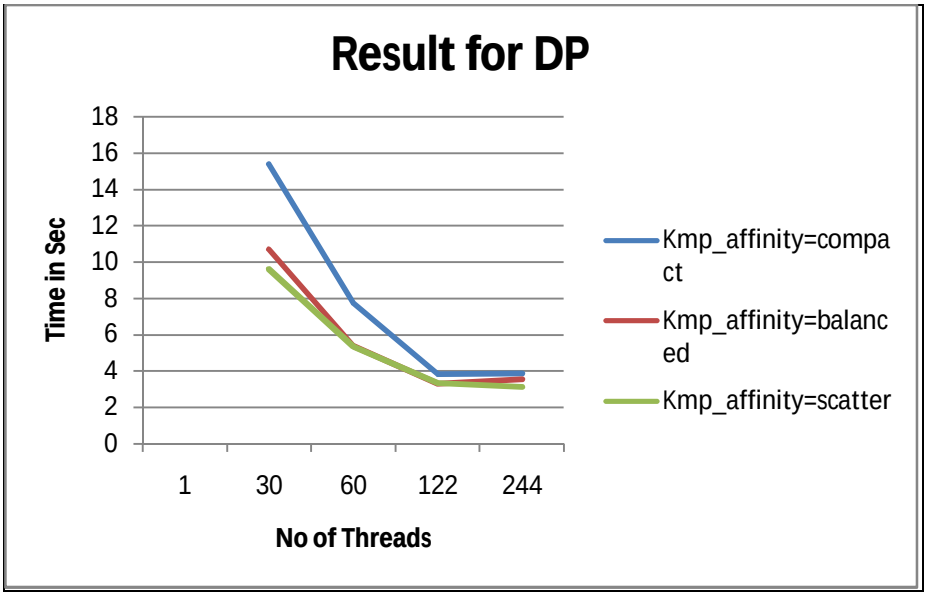

Fig 7 Result for Double Precision on Intel Xeon Phi

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

V. CONCLUSION

The aim of this approach is to address all issues and hurdles related to applications performance and achieve the highest possible performance on Intel architecture, while utilizing all the possible parallel execution resources. It is clear that the different modes of programming for the Xeon Phi are important for the applications. The division of work between the host and the Xeon Phi card is critical for the applications. Native mode uses the Intel Xeon Phi card only without the host (CPU). We demonstrate a methodology of sharing code between traditional Intel Xeon and Intel Xeon Phi architectures, and hence a suitable development path for the acceleration of legacy HPC codes.

VI. FUTURE SCOPE

In future the code can be port in offload mode on Intel Xeon Phi coprocessor. Also try to convert this code into Message Passing Interface (MPI) that uses distributed memory architecture. If the code is converted into MPI, we can use the symmetric mode option to run the code on both the architecture.

REFERENCES

1. Simon Heybrock, Blint Jo, Dhiraj D. Kalamkar, el.at, “Lattice QCD with Domain Decomposition on Intel Xeon Phi Co processors", ICHPC 2014 IEEE DOI 10.1109/SC.2014.11.
2. Eisenlohr J, Hudak DE, Tomko K, Prince TC. “Dense Linear Algebra Factorization in OpenMP and Cilk Plus on Intels MIC Architecture: Development Experiences and Performance Analysis"; TACC-Intel Highly Parallel Computing Symposium 2012.
3. Jo B, Kalamkar D, Vaidyanathan K, Smelyanskiy M, Pamnany K, Lee V, Dubey P, Watson I, William “Lattice QCD on Intel Xeon Phi Coprocessors". Supercomputing, ser. LNCS. 2013;7905:4054
4. K. Vaidyanathan, K. Pamnany, D. D. Kalamkar, A. Heinecke, B. Joo, el at. “Improving Communication Performance and Scalability of Native Applications on Intel Xeon Phi Coprocessor Clusters,"in IPDPS 2014 (28th IEEE International Parallel and Distributed Processing Symposium).
5. https://en.wikipedia.org/wiki/Monte_Carlo_method
6. http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html
7. http://www.endtoend.in/ete2011/INTEL/Tech-Phi-rive/Portal/index.html
8. http://www.hpctoday.com/hpc-labs/performance-portable-accelerator-programming-in-fortran/
9. https://software.intel.com/en-us/articles/case-study-achieving-high-performance-on-monte-carlo-european-option-using-stepwise