International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May 2018 www.irjet.net p-ISSN: 2395-0072

Performance Analysis of Combined symmetric and Asymmetric Parallel Programming Method for IoT Application

Elezabeth Thomas[1], Saju A[2]

[1]*Student, Dept. of EC Engineering, Believers Church Caarmel Engineering college, Kerala, India*
[2]*Professor, Dept. of EC Engineering, Believers Church Caarmel Engineering college, Kerala, India*

Abstract - *Parallel computing was present since the early days of computing. In high performance computing the important parts are parallel algorithm and simulation. The execution may require lots of power for processing. For the analysis purpose best area chosen are SMP and ASMP. With exposure to SMP and ASMP in different way, the Project was built using Raspberry Pi 2 Model B embedded processor. For SMP excellent platform selected is a computer with 4 core processor as single board i.e, Raspberry pi 3 Model B hence memory concept used in it is shared memory. For ASMP, using a Wi-Fi network we provide server cluster connection using two Raspberry Pi board. Sobel Filter is used in image processing and computer vision, particularly within an edge detection algorithm where it creates an image edges were chosen as a target application to analyze the performance. Sobel filter is the best target application which having two dimensional array turned out the computation of edge by these platforms. Also perform hybrid verses SMP and ASMP analysis of k-mean algorithm for clustering a group of data. Finally reaching at a conclusion that hybrid system, achieves the better performance than SMP and ASMP Programs*

***Key Words*: SMP, ASMP, sobel filter, K-mean Algorithm, Hybrid network**

1. INTRODUCTION

Parallel computing is a type of compilation in which many calculations are performed at the same time. Basic principle of parallel computing is that compilation can be divided into smaller sub problems each of which can be solved simultaneously.

Parallel computing was present since the early days of computing. In high performance computing the main area of focus are the parallel algorithm & simulations, it may require a lot of power for operating. For regular use we have to increase the clock frequency of CPU, so that CPU could execute high number of instructions for second.

Frequency scaling is no longer possible after a freezing point because the power require for processor starts to go nonlinear this is known as Power Wall. Multiple CPU cores providing was solution found by the vendors instead of increasing the clock frequency of CPU processor and in which each chips capable of executing separate instruction streams. Parallel computing provides computational power when sequential computing cannot do so. But parallel programming is some more difficult than sequential programming. Mainly because of these reason:

1.Dividing sequential computations into parallel computation can be complex or even impossible.
2.Due to different errors resulting in the computation may cause program correctness more difficult.
3. Speed up is the only reason why we bother paying for this complexity.

Main advantage of parallel programming is that increase in program performance; and it is expected to be faster than the sequential program.

Parallelism & concurrency are closely related concepts. In case of parallel program, parallel hardware to execute computation more quickly. Efficiency is its main concern to make the parallel hardware to obtain optimal speed. For this different techniques are available.

Multiple executions are may not performing at the same time for a concurrent programming. Which also improves modularity, responsiveness along with providing better maintainability. Where in this type of program execution when an execution start & how it shares the computational results are important. Parallelism manifest itself at different granularity level like bit level.

- Bit level parallism in which processing multiple bits of data in parallel
- Instruction level parallism in which executing different instruction from the same instruction stream in parallel.
- Task level parallism where separate instruction stream in parallel

Parallel computers can be classified by considering the level at which the hardware supports parallelism, mainly

1.1 Symmetric Multi-processing

One of the main concept for shared memory multiprocessor architecture is symmetric multiprocessing(SMP). Where bus providing a medium for interconnecting cache of each processor in the SMP. In a bus based structure access time for all memory locations is equal because all the access time for all processor gets data from the main memory through the bus.

In any multiprocessor, main memory access is a bottleneck; to reduce the memory demand of a processor multilevel caches are introduced. To share the memory bus between more processor become possible by introducing a concept of multilevel cache.

For a system when the number processor increases contention for the bus also increases. One of the main way-out may be to use switches (crossbars, multistage networks, etc.) instead of a bus. Hence the scalability of the SMP model restricted but while using switch given a parallel point to point connections, also use which may cause the implementations of cache coherence difficult.

An important problem with shared memory is coherence; i.e. when the shared data are cached which may replicate in multiple caches. Different processors having the data in the cache memory may become inconsistent. Multi cores on the same packaging will execute different threads, when there is no thread to execute it will be switched off. The Multi core system can be dual, core etc.

To overcome cache coherence problem by dynamically recognize any potential inconsistency at run-time & carry out preventive action. There by consistency maintenance becomes transparent to programmers, compilers, as well as to the operating system.

Two different modes of programming challenges remain in the symmetric multi-processing; one for the CPUs itself and one for the interconnect between the CPUs. A single programming language used in an architecture would have to be able to not only to comprehend the memory locality, but also, partition the workload which is the one of the important criteria in mesh-based architecture.

To implement the shared memory system, the library now a day used is OpenMP. In symmetric multi-processing(SMP) for shared memory parallel application are performed by OpenMP(open Multi Processing). It a library consists of compiler directives and library routines for parallel computing programs. All the memory threads of same parallel program will be sharing same address space.

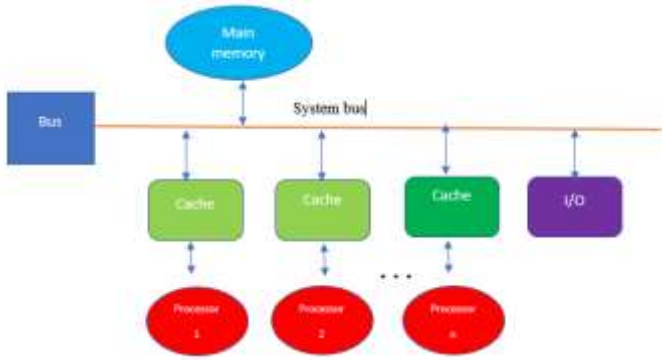

Fig1: Diagram of symmetric multi-processing system

Because it is not a complex language, and we don't have to spend lot of time to do parallel programming using Open MP. Open MP supports fortan & C++ and C.

OpenMP helps to interface for developing parallel computing application from standard desktop computer to super. It is an implementation of a master node which perform the execution of multithreading task by guiding a number of slave node and parallel computing used for diving threads among them for the efficient computation.

1.2 Asymmetric Multi-processing

For handling multiple CPUs **Asymmetric multiprocessing** (**ASMP**) was the only method available and after that the concept of symmetric multi-processing arrived. It has also been used to provide less expensive options on systems where SMP was available. In embedded system the individual task is very important to perform and ASMP was useful for such a dedicated individual task in that platform. It is not important that all CPUs are not treated equally in a ASMP; i.e. some system that only allow (either at the hardware or software) one CPU to perform software code or may only allow one CPU to perform general input output operations. Other ASMP systems could enable any CPU to execute software code and to perform input output operations

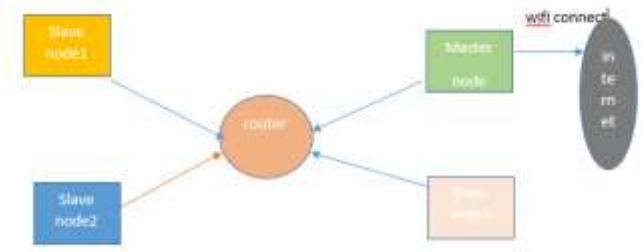

Fig2 : Diagram of asymmetric multi-processing system

Message passing model parallelism is achieved by having many processes co-operating on the same task. Each process has access only to its own data that is all variable used in this method is private. And each processes communicate with each other by sending and receiving messages. A message transfer mechanism in which a number of data items of certain type from the memory of one process to the memory of another process. A message typically contains ID of sending processor, ID of receiving processor, the type of data items, the number of data items, the data itself, finally a message type identifier. It doesn't have any proper structure that's why it is well fitted with object oriented language like Java, C++. Generally, a message passing can be either synchronous or asynchronous. A synchronous send is not completed until the message has gone. An asynchronous send completes as soon as the message has gone. In MPI the receiving messages is always synchronous. High Performance Interconnect or a Memory Controller are not required for this mechanism because it is not a complicated hardware and also multi computer used in this mechanism is less costly. Beowulf clustering is chosen as architecture for message passing in multi computer between the different nodes. The system contains master node or server node which guide the one or more slave node for task execution via Wi-Fi or Ethernet or some other network. Beowulf clustering architecture uses software like MPI, Unix, PVM etc.

In this work for implementing asymmetric multi-processing MPI (Message Passing Interface) was the technology selected.

It is a library which standardized and rather written in languages like C, C++, Fortan and implementation can be performed via openMPI, MPICH, MVAPICH. Also for python programmers MPI can be introduce by the implementation of Mpi4py.

2. SOBEL FILTER

Output regions of image that given to a sobel filter in which high spatial frequency that correspond to edges of the image. When an input gray scale image is given it will immediately find the absolute tilt magnitude.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Fig3:sobel matrix values for each co-ordinate

The kernels are designed to acknowledge maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The input image can be executed by kernel separately, to produce separate values of the tilt component in each segments

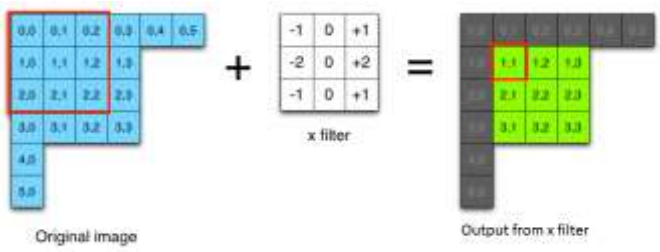

Fig4: sobel filter operation of x filter with original image

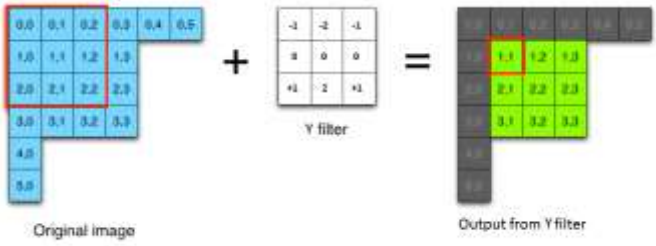

Fig5: sobel filter operation of y filter with original image

The absolute magnitude and orientation of the tilt can be find out by combining the these two at each point and the orientation of that tilt. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

Typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

In case of SMP, the two dimensional Image array given to shared memory processor and assign them to each thread. In case of ASMP, for performing convolution of a pixel in ASMP we need the pixels surrounding it. Check figure 4, the output pixel 1,1 is calculated by the convolution which involved all the pixels surrounding it. Either along y axis or x axis we can divide the image array symmetrically as shown in Figure5 and sent them to different cluster nodes, the edges in the image needs pixels from the previous row which is not local to that particular. Last and first row of first and third chunks can be used to perform edge detection by the first and last row of second chunk respectively

3. K-MEAN CLUSTERING

Clustering is a type of dividing a set of data into different group so that the object within the group share a common character. In the case of clustering there is no guidance of how do you do group. So it is given by the name of unsupervised learning. Once we form the group label the group. Also given by the name learning by Observation. Any good clustering approach will generate high quality clusters. High quality clusters means there is high intra cluster similarities which means the object within the group are

tightly bound or similar to each other compare to the objects between the clusters. The quality of the cluster developed during the clustering mainly depends on the similarities we used and how it is implemented etc.

Partition clustering approach it takes the data base of n objects and create k partition or k cluster and the basic focus of here is to reduce the sum of squared distance. If a cluster is represented by centroid or medoid partition clustering approach finds out for each of the cluster what is the distance between a object within the cluster and cluster or centroid or medoid it takes the square of the distance finds the sum of square of distance for each of the distance then each of the k clusters. Global optimal is one the partition criteria for generate all partitions and try to find the best one.k-means and k-medoid clustering algorithm. These two clustering techniques mainly tries to reduce the sum of the squared distances using the Heuristics approach.

K-means clustering is a simple and understandable clustering method, which divide a set of data observations into k cluster. In k-mean clustering algorithm one of the important term is centroid which is actually the mean of cluster and in K-medoid algorithm one of the object in the cluster represents each of the cluster. It has many advantages in the field like computer vision, agriculture, image and market segmentation. Its huge applications and its simple measurement complexity make k-means clustering as one of the best method now a day.

When the dimension d>1 and cluster number become k>1 it is hard to discover minimum cost function of the k-mean clustering. Scientists came up different solution but still it is difficult to measure. So we want to build a parallel version of a k-means method on a clustering which having high accuracy and also increases computation speed.
The algorithm works as follows:

1.First we initialize k points as centroid(mean) by random manner.

2.We classify each data to the closest centroid by Euclidean distance method and update new cluster center by taking minimum value from the calculation.

3.We repeat the steps until number of iteration have same calculation value.

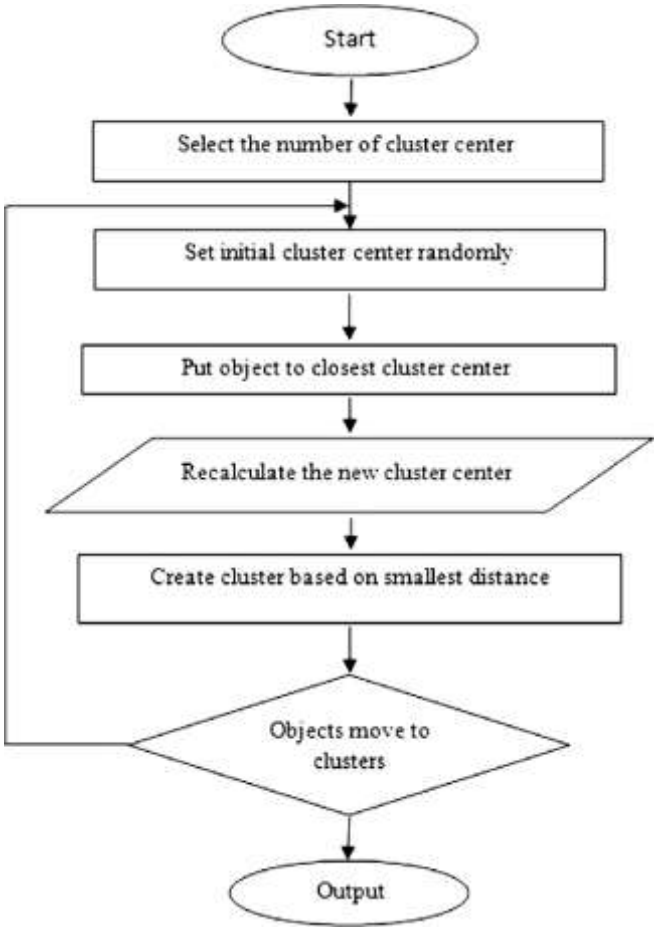

Fig6: flow chart of k-mean clustering algorithm

SYMMETRIC MULTIPROCESSING USING OPENMP

Parallelization using multiple cores of the same CPU can be introduced into a C/ C++ program by using constructs from the standardized OpenMP library. OpenMP should come pre-installed in a system with any standard version of gcc (GNU C compiler) or g++ (GNU C++ compiler) on both old and newer version of *nix based systems, where *nix can mean any Unix or Linux derivatives. These constructs can be used in any C/C++ program by including “omp.h” header or pre-processor directive directly in the program and using the compiler flag -fopenmp when compiling using gcc / g++ to instruct the compiler to include OpenMP during the link phase. The compiler flag is required as OpenMP is developed as a set of code transforming pragmas which are only applied at compile time. The most resource-demanding procedures or sub-routines in the program are to be parallelized for parallelisation to have maximum effect on the program execution time. Hence sobel_filtering sub-routine was chosen to be parallelized using OpenMP. “omp parallel” compiler directive instructs the compiler to parallellize the code contained inside this construct using

parallel threads running in different cores of the same processor. In this case, the values corresponding to the pixels involved in the convolution will be sent to the different cores. Nested loops can be collapsed using

#pragma omp parallel for collapse (2)

clause where the number in parenthesis instructs the compiler to collapse the first two for loops.

Sharing processing or computation load among different cores of a processor is expected to decrease the program execution time compared to when running a sequential program with a single core. However, this is dependent on the 'edge cases' during parallelisation problem mentioned elsewhere and on the way the program is implemented.

When the case of Kmean clustering as we know that there are two steps where one of the step such as E-step can done only by openMP method. When M-step is used by direct openMP parallelization which may result WAW(Write-After-Write) i.e. different data points may add to the same cluster. So using openMP we only focus on E-step.

When the number of core increases M-step is not too much time consuming but when E-steps scales well M-step become slow.

ASYMMETRIC MULTI-PROCESSING USING MPI

In asymmetric multiprocessing using mpi, the data is split based on the number of nodes available. This way, the master node divides and sends the load among all slaves nodes. Lower number of nodes in the Beowulf Cluster can result in larger chunks being sent to each node to be processed locally. This places a greater demand on the processor and on other resources native to the node. Smaller chunks of data will be communicated among the nodes as the size of the network increases. Also, ideally, the computational capability of each node should also be considered when distributing the load. Our implementation uses nodes with similar computational power and hence data is distributed as equally-sized chunks.

In a setup where different nodes have varying processing capability, the computational capability of slave nodes may be assessed and communicated to the master node so that larger chunks of data are distributed to more capable nodes and smaller chunks to less powerful nodes.

The simplest way for this is to assign ranks to nodes based on their computational capability or willingness to allocate resources for the parallel process. Assessing processing power of nodes on-the-run can be an attractive alternative method, but can introduce a greater demand on resources as it requires more information such as the underlying processor and resources supporting it to be shared. The slave nodes and the master process the individual data chunks locally. Slaves communicate the processed data back to the master node. The master node communicates to the slave node either through collective communication constructs or using point to point communication constructs. Slaves transfer the processed information back to the master using point to point communication constructs. MPI implements collective and point to point communication procedures as blocking constructs. It is left to implementations to avoid deadlock conditions.

The master node takes care to communicate the chunks of data such that data near the edges of the chunks constitute a section of the data from adjacent chunks so that the data overlapping can be achieved in a way that convolution operations can be performed without error. Further, additional overhead in communication, although minimal, is unavoidable as each node should have information such as the total size of the image before processing so that memory can be allocated dynamically i.e., on the run. Additional information related to application too are to be communicated.

- Data split based on number of nodes available
- Master node splits data into chunks among nodes and communicates the chunks among the nodes
- Slave nodes and the master process the individual data chunks locally. Slaves communicate the processed data back to the master
- Master node communicates to slave nodes either using collective communication constructs or using point to point communication constructs.
- Slaves communicate back to the master using point to point communication procedures.
- MPI implements collective and point to point communication procedures as blocking constructs. It is left to implementations to avoid deadlock conditions.
- The master node takes care to communicate the chunks of data such that data near the edges of the chunks constitute a section of the data from adjacent chunks so that the data overlapping can be achieved in a way that convolution operations can be performed without error.
- Further, additional overhead in communication, although minimal, is unavoidable as each node should have information such as the total size of the image before processing so that memory can be allocated

dynamically i.e., on the run. Additional information related to the application too are to be communicated.

In parallel K-mean clustering distribution of data to different process is carried by MPI-Bcast, and to transfer information whenever it is needed using the command MPI-Allreduce. In this process with any further complication E-step and M-step can be executed. In this case speed is better than openMP.

HYBRID of SMP and ASMP

As both SMP and ASMP programs were built successfully, a hybrid of both the models was built by adding openMP constructs to the existing ASMP program. The Images which are divided symmetrically at the master node are sent to slave nodes for local processing. In each node being a Raspberry pi, which itself can act as a perfect SMP as described in section 5, openMP parallelization was applied locally.The reason why the SMP and ASMP setup is able to achieve high speedup is because, 16 Cores are processing different parts of the image concurrently, resulting in high speed up

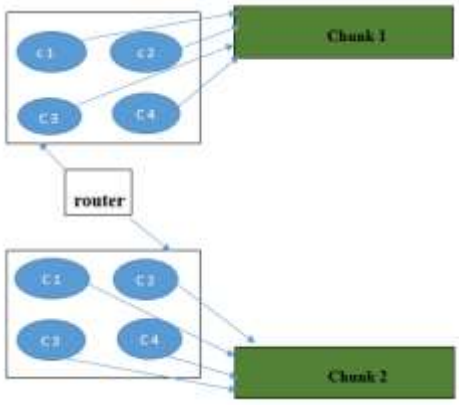

Fig7: Block diagram of hybrid system

Also, The Intercommunication Latency which is low due to 10/100 Mbps Ethernet connection between the Nodes helped to reduce the overhead associated with the Message Passing. Since all the nodes involved in the Cluster uses same Processor and of same architecture, there was no difficulty or issues when implementing it. The time taken for the same image used for discussion in all above sections was 0.02 seconds, which is clearly a better performance than both SMP and ASMP programs

In K-mean clustering for a hybrid version we mix-up openMP programming codes to the MPI programs. This time we have to test different combinations of MPI and openMP task. Finally we get the result as speed is increased for a hybrid version for a number of openMP and MPI thread.

4. RESULT AND CONCLUSION

From SMP, ASMP and Hybrid applications the performance time was find for four different times and an mean time was calculated to compare the performance of different implementation by the target application. Images of two different size were selected to prove the point that large data can be processed efficiently by programming parallel hybrid concept instead of sequentially. This helps to exploit the availability of multiple cores in the processor system and also provides an opportunity to getting performance in distributed system in case of IoT(Internet of Things).

Image pixels	Sobel Sequential (time in s)	Sobel OMP (time in s)	Sobel MPI (time in s)	Sobel Hybrid (time in s)
256 x 256	0.2123	0.0851	0.0912	0.0520
	0.2312	0.0755	0.0822	0.0500
	0.2142	0.0721	0.0773	0.0433
	0.2134	0.0763	0.0721	0.0500
256 x 256(avg)	0.2178	0.0772	0.0807	0.0488
512 x 512	0.8422	0.2432	0.2342	0.1001
	0.8214	0.2531	0.2256	0.1030
	0.8351	0.2821	0.2672	0.1000
	0.8121	0.2012	0.2238	0.1025
512 x 512(avg)	0.8277	0.2449	0.2377	0.1014

Performance evaluation of the above details plotted in graphical manner which include the variation in time required to execute the sobel filter in openMP, MPI and Hybrid is given below for different image size of 256x256 and 512x512

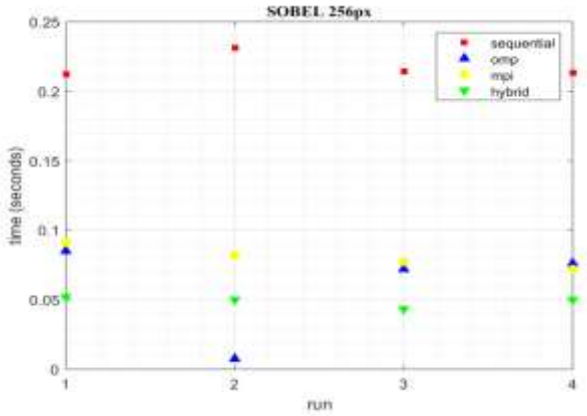

Fig8: performance analysis of sobel filter for an image size of 256px

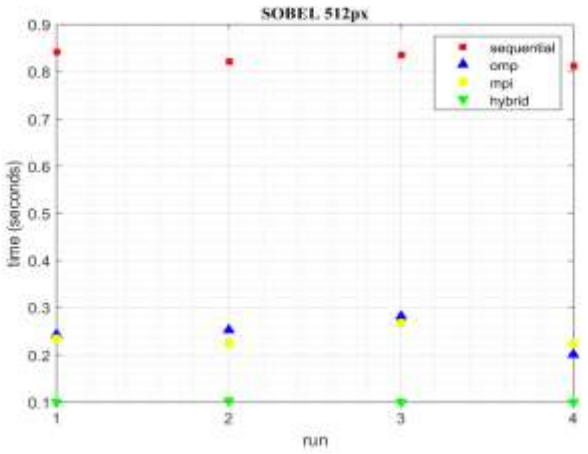

Fig9: performance analysis of sobel filter for an image size of 512px

Also from the detailed study of set of input data of different category species are classified using k-mean clustering in openMP, MPI and Hybrid is given below.

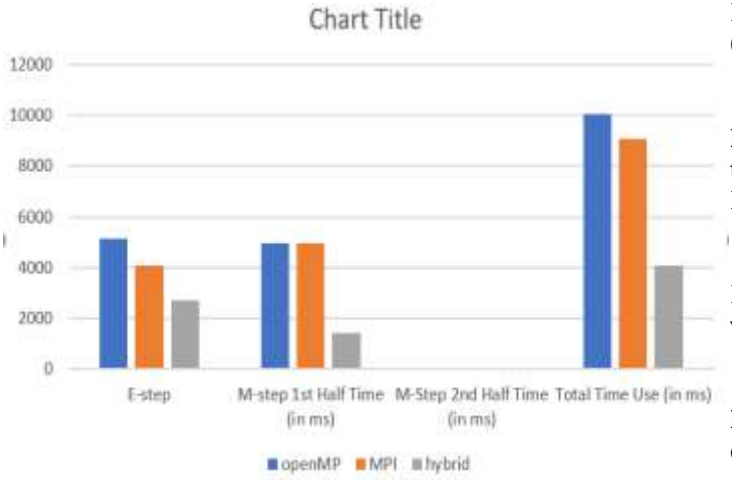

Fig10: performance analysis of k-mean clustering for input data collection

Hence the distributed system in parallel computing with hybridization of SMP and ASMP can reduce the time required for the edge detection process by a factor 40-50%.

REFERENCES

[1] **N.E.A.Khalid , S.A.Ahmad, N.M.Noor, A.F.A.Fadzil and M.N.Taib** (2011), Parallel approach of Sobel Edge Detector on Multicore Platform , International journal of Computers and Communications Issue 4, Volume 5, 236-244

[2] **Michael Lescisin, Qusay H. Mahmoud (**2016), Middleware for Writing Distributed Applications on Physical Computing Devices, IEEE/ACM International Conference on Mobile Software Engineering and Systems, 21-22

[3**] Michael Lescisin and Qusay H. Mahmoud (2017)**, DCM: A Python-based Middleware for Parallel Processing Applications on Small Scale Devices, IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)

[4] **Andrew K. Dennis** (2013), Raspberry Pi Super Cluster, www.packtpub.com

[5] **Nazleeni Haron, Ruzaini Amir, Izzatdin A. Aziz, Siti Rohkmah Shukri, Low Tan Jung** (2010), Parallelization of Edge Detection Algorithm using MPI on Beowulf Cluster, Innovations in Computing Sciences and Software Engineering, 477-478

[6**] Poman P.M. So and Wolfgang J.R. Hoefer** (2001), Poman P.M. So and Wolfgang J.R. Hoefer, IEEE MlT-S Digest, 2007-2010

[7] **Honggang Wang, Jide Zhao, Hongguang Li and Jianguo Wang** (2008), Parallel Clustering Algorithms for Image Processing on Multi-core CPUs, International Conference on Computer Science and Software Engineering, 450-453

[8] **Azhar Rauf, Sheeba, Saeed Mahfooz, Shah Khusro and Huma** Javed (2012), Enhanced K-Mean Clustering Algorithm to Reduce Number of Iterations and Time Complexity, Middle-East Journal of Scientific Research 12 (7): 959-963

[9] **Murtagh** (1983), A Survey of Recent Advances in Hierarchical Clustering Algorithms, the computer journal, vol. 26, NO. 4

[10] **Vincent D. Blonde, Jean-Loup Guillaume, Renaud Lambiotte and Etienne Lefebvre (**2008), Fast unfolding of communities in large networks, physics.soc-ph, 1-12

[11] **Tayfun Kucukyilmaz** (2014), Parallel K-Means Algorithm for Shared Memory Multiprocessors, Journal of Computer and Communications, 15-23

[12] **V.Ramesh, K.Ramar, S.Babu,** Parallel K-Means Algorithm on Agricultural Databases, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 1, January 2013, 710-713

[13] **K. A. Abdul Nazeer, M. P. Sebastian** (2009), Improving the Accuracy and Efficiency of the k-means Clustering Algorithm, Proceedings of the World Congress on Engineering Vol I WCE 2009, July 1 - 3, 2009, London, U.K.

[14]**Ujjwal Maulik and Sanghamitra Bandyopadhyay** (2012), Performance Evaluation of Some Clustering Algorithm and validity indices, IEEE transaction on pattern analysis and machine intelligence, vol.24, no.12, 1650-1654

[15] **Sanpawat Kantabutra and Alva L. Couch** (2012), Parallel K-means Clustering Algorithm on NOWs, Technical Journal, vol.1, no.6

[16] **Domenico Talia** (2002), Parallelism in Knowledge Discovery Techniques, J. Fagerholm et al. (Eds.): PARA 2002, LNCS 2367, pp. 127–136

[17] **Dr.Urmila R. Pol** (2014), Enhancing K-means Clustering Algorithm and Proposed Parallel K-means Clustering for Large Data Sets, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 5, May 2014, 1489-1492