International Research Journal of Computer Science (IRJCS)
Issue 06, Volume 4 (June 2017)
ISSN: 2393-9842
www.irjcs.com

Programming Modes and Performance of Raspberry-Pi Clusters

Anil Jaiswal
Research Scholar, Department of Computer Science,
Sant Gadge Baba Amravati University, Amravati,India
jaiswal.anil@gmail.com
Dr. Sanjeev Jain
Vice-Chancellor
Shri Mata Vaishno Devi University, Katra, J &K
sanjeevjain@yahoo.com
Dr. V. M. Thakre
Professor and Head, Department of Computer Science
Sant Gadge Baba Amravati University, Amravati,India
vilthakare@yahoo.co.in

Manuscript History
Number: IRJCS/RS/Vol.04/Issue06/JNCS10081
Received: 09, May 2017
Final Correction: 23, May 2017
Final Accepted: 29, May 2017
Published: June 2017
Citation: Jaiswal, A., Jain, D. S. & Thakre, D. V. M. (2017). Programming Modes and Performance of Raspberry-Pi Clusters. research report published doctoral dissertation , Sant Gadge Baba Amravati University, Amravati,India.
Editor: Dr.A.Arul L.S, Chief Editor, IRJCS, AM Publications, India
Copyright: ©2017 This is an open access article distributed under the terms of the Creative Commons Attribution License, Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Abstract— *In present times, updated information and knowledge has become readily accessible to researchers, enthusiasts, developers, and academics through the Internet on many different subjects for wider areas of application. The underlying framework facilitating such possibilities is networking of servers, nodes, and personal computers. However, such setups, comprising of mainframes, servers and networking devices are inaccessible to many, costly, and are not portable. In addition, students and lab-level enthusiasts do not have the requisite access to modify the functionality to suit specific purposes. The Raspberry-Pi (R-Pi) is a small device capable of many functionalities akin to super-computing while being portable, economical and flexible. It runs on open source Linux, making it a preferred choice for lab-level research and studies. Users have started using the embedded networking capability to design portable clusters that replace the costlier machines. This paper introduces new users to the most commonly used frameworks and some recent developments that best exploit the capabilities of R-Pi when used in clusters. This paper also introduces some of the tools and measures that rate efficiencies of clusters to help users assess the quality of cluster design. The paper aims to make users aware of the various parameters in a cluster environment.*

Keywords *— Raspberry-Pi Clusters; Hadoop; MapReduce; MPI; openMPI; benchmarking clusters;*

I. INTRODUCTION

Parallel and distributed computing technology focuses on maximizing parallel computing capabilities inherent in multicore processors and their capability for interaction through networking (Culler et al., 1998; Hyung et al., 1997; K. M. Lee & Lee, 2012; S. W. Lee et al., 2005;

© 2014- 17, IRJCS- All Rights Reserved

Sodan, 2005). Such hardware and software architectural combinations help improve speed, volume and efficiency of computing resources. Some of the most accepted resources include the following: single instruction multiple data (SIMD) graphics processing unit (GPU),and general purpose graphics processing unit (GPGPU); simultaneous multithreading (SMT) non-uniform memory access (NUMA), symmetric multiprocessor (SMP) architecture, architectures, and superscalar processor [1]–[6].

Similarly, software developers have designed applications, platforms and layouts to fully exploit the capabilities of a hardware configuration, initially meant for specific purposes. Such software technologies, often available as open-source enable users to extract parallel, distributed, and concurrent computing for various [6], [7].
As there are many different frameworks of parallel and distributed computing, it would be of great help to have performance comparison studies for the frameworks we may consider. Typically, clusters are workstations, or personal computers as 'nodes' connected to servers that collect, monitor, assess, and distribute tasks before collecting the results and aligning them from the nodes to display the results. Such systems are finding increasing relevance in the contemporary age of huge amounts electronic data and requirement of computational prowess and speed of processing electronic data. Users with disparate interests access such results or data to enable decision-making in a variety of applications ranging from wide areas of research, socio-economic and commercial activities. Amongst the most important considerations to accessibility of such configurations are apparently systems, costs, and energy requirements.

Raspberry-Pi

The Raspberry Pi (R-Pi) allows for creating a low-cost, low energy, portable cluster that attends to the needs of small users (that would not have the means or access to mainframes, servers). Generally, one R-pi would be assigned the task of the main server (alternatively, one could assign multiple servers for efficiency) and required number of nodes (ranging from 4 to 128 (not a limiting factor), according to the need) added to it to form the cluster. One of the advantages of such clusters is the direct control over each node, in turn implying the advantage that failure of a node does not affect the computation or task assigned to the cluster. Consequently, nodes can be independently programmed to interact with each other, in addition to receiving and transmitting data and instructions from and to the server. The basic mechanism that enables implementing such clusters is the 'parallel processing' capability. This allows users to build, test, and deploy such clusters easily for students and enthusiasts remotely. Users can then use mainstream tools such as MPI and Hadoop on such clusters and access the range of services offered by commercial, large-scale servers and cloud-computing facilities offered by larger data-handling- frameworks.

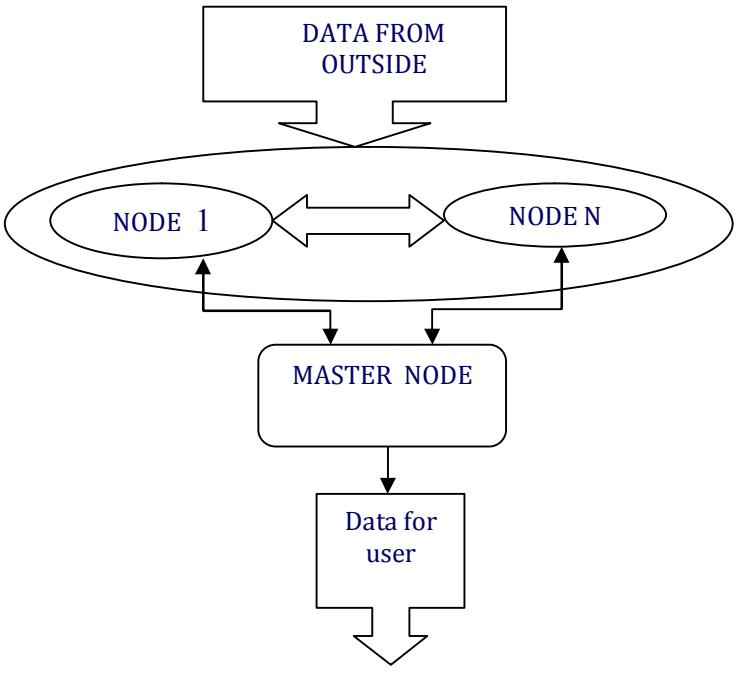

Fig. 1. Flow in a Hadoop Cluster

© 2014- 17, IRJCS- All Rights Reserved

Thus, R-Pi clusters are miniature versions of supercomputers that compute, access, and disseminate (computation, storage, and transmission) large amounts of data. The underlying features binding the capabilities of supercomputers are multicores, multiprocessors, and parallel computational technologies that are evolving continually and creating more powerful, faster, and efficient machines. Quantitative comparison of R-Pi and supercomputer features reveal a mixed set of outcomes that need to weighed-in judiciously before choosing the right combination and array for an assigned task. For example, whereas in a 12-core Intel Sandy Bridge-EP the energy and cost measures 346 MFLOPS/Watt and 21.30MFLOPS/$, and 247.2/Watt and 21.60/$ in a 16-core AMD Opteron 6376 the comparable values for R-Pi computer is less than 100MFLOPS/Watt and @5MFLOPS/$ [8]. Obvious disparities appear in the energy and the cost efficiencies.

However, the trade-off is between speed of operation and Memory capabilities of R-Pi clusters, owing to their low cost of operation, users have to account for operating capabilities and speeds. These measuring techniques vary by different factors giving rise to a need for benchmarking standard. A benchmarking procedure becomes crucial owing to the large amounts of data storage and computation in a HPC (High Performance Computing) system. A typical R-Pi cluster would comprise of a *Master* R-Pi monitoring an array of slave or nodes that in turn can interact with each other. The server R-Pi executes the tasks required by HDFS and thereby operates as a manager of NODES under it. The Hadoop-run NODES have a job tracker in addition to a task scheduler. Once a node completes an assigned task, it becomes free to execute the next task in the queue.

Fig. 1 above shows typical interconnections and interactions of clusters mounted with Hadoop. As depicted, each NODE (numbered 1 to N) can communicate with other nodes in addition to receiving directions and task assignments from the MASTER R-Pi. The Nodes collect data from the outside environment, process it according to directions received from the MASTER and send the data in the required format. The data from the nodes pass through Map Reduce operations before passing it to the Master. The MASTER finally consolidates the data and makes it available to the user. The superficial practical networking required in clusters suggests that the transmission of data could be a possible cause of latency issues. A practical cluster may have multiple second-level Masters in turn controlling a 'thread' of nodes. Such threads could be comparable to the 'cores' in a multiprocessor that can run programs or program segments in parallel or concurrently. The Second level Masters would then be controlled by a First-level MASTER.

Such a system replicates the multicore, multiprocessor architecture in parallel processing architectures found in larger CPU's. Each node in the network performs only one task at a time, thus improving the power and speed of computation. In addition, deploying such an architecture ensures a fault-tolerant system, that is also scalable [9]**.** Since the R-Pi was designed to handle data from external environment (sensor), its internet connectivity speed and bandwidth capability is limited to serve such purposes. It is not meant essentially for heavy computation or data storage tasks. Expectedly, a single core R-Pi does not perform well on these metrics. In order to overcome these lacunae in its computing capabilities, studies have attempted and successfully implemented R-Pi clusters that imitate multiprocessing capabilities of supercomputers, though in a limited way. However, given the range of hardware limitations they conform adequately to, the challenging requirements of low-end users making them an accessible, low-cost, and power efficient alternative. In essence, R-Pi works well with standards such as the MPI and Hadoop (owing to its Map-Reduce capabilities).

II. LITERATURE REVIEW

A. Hadoop

Hadoop deployed on an R-Pi cluster, can result in significant improvement in the functionality and computational prowess of the Framework. Amongst the earliest exponents of Installing Hadoop on an R-Pi was Jamie Whitehorn. The author sought to introduce the concept of distributed architecture to students through such deployment. He inferred that students can better appreciate the features of Hadoop by first working on such smaller systems. The method also carries the advantage that each node in an R-Pi can be controlled independently [10]. In another Hadoop implemented scheme, the researchers used the Map-reduce functionality and DFS (distributed file system) to enable nodes in a cluster to collect, manage and transmit sensor data to a master node on demand. [11].

Following the work by [12], Shaun Franks and Jonathan Yerby, [13], used the initial design of a R-Pi based supercomputer structure to implement a low-cost high-level data management. The work demonstrates a fault-tolerant Hadoop-based R-Pi cluster to handle Big Data [14]. Mahesh Maurya and Sunita Mahajan have demonstrated added capabilities of an R-Pi cluster when run on Hadoop [15]. The authors sought to exploit the functionalities of Map-reduce algorithms further, through 'wordcount', 'pi', and 'grep'. Researchers Sekiyama and colleagues (2015) demonstrated the capability of Hadoop on R-Pi in their portable IOT implementation that uses Hadoop as the database.

© 2014- 17, IRJCS- All Rights Reserved

Research in this area continues to exploit the utility of R-Pi cluster as a development tool in increasing domains as a low cost, low energy and fault-tolerant structure that replicates the capabilities of a supercomputer that is both accessible and portable [16].

B. MPI and openMP

The parallel processing architecture fall under two broad types:

Symmetric Multiprocessing: Here each node is an independent peer and the Physical memory is shared amongst all the nodes in the cluster. This arrangement does not have a typical Master-node hierarchy.

Asymmetric Multiprocessing: In this arrangement, the Master-node architecture is deployed, where each node is assigned a specific task by the master and thus controls the task assigned to the cluster.

The methodology of Memory Access between nodes follows either of the two following formats: NUMA (Non Uniform Memory Access) and MPI. In NUMA, the time for access to memory depends on the proximity of the node to the Memory location. The nodes access the Memory through high-speed interconnection. In MPI, that is the widely preferred clustering architecture owing to its accessibility, cost, and ease of use, memory sharing can take place only through mutual consent, not directly between nodes.

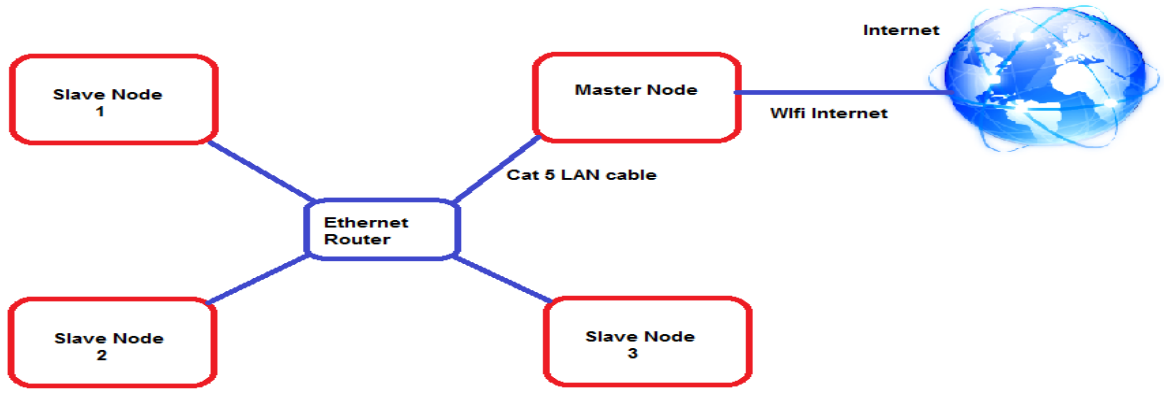

Fig. 2 Message Passing model (Beowulf Cluster) [17]

The Message Passing Interface is the standard communication between and amongst different processes. Effectively it allows for parallel processing of R-Pi's in a Beowulf cluster. This platform makes it possible to use the smaller processors to run on lighter software's and using standard Ethernet adapters. Thus, it can be deployed by enthusiasts and learners easily without the need for specialized, dedicated hardware. In addition, Beowulf clusters use open-source OS software: Linux, FreeBSD, or Solaris and PVM (Parallel Virtual Machine) or MPI.

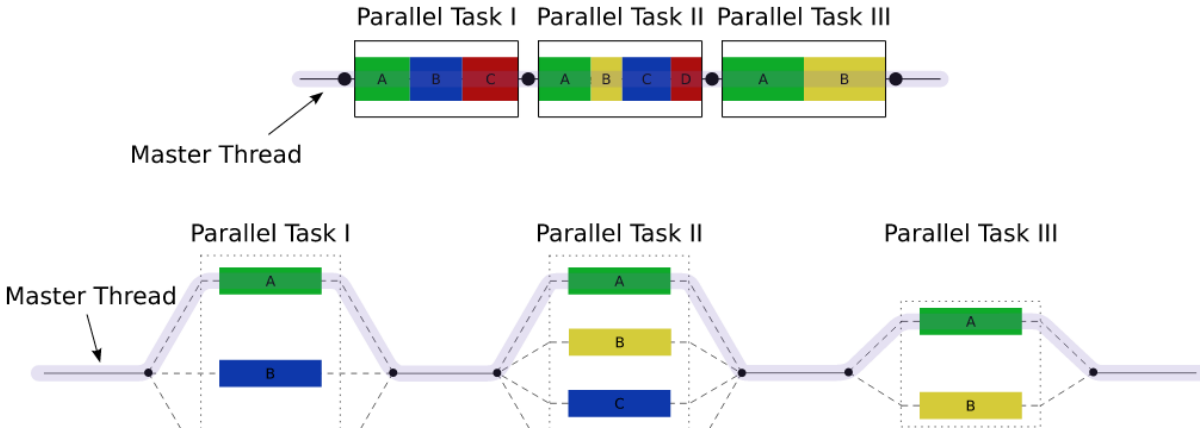

Figure 3 openMP Workflow Visualization [17]

openMP is a dedicated technology for SMP (Symmetric Multiprocessing) that offers the users the advantage of allowing for using shared Memory. It has in-built compiler directives and library routines that allow for parallel processing. All threads in a parallel program share same memory addresses (achieved by same base for memory mapping).

© 2014- 17, IRJCS- All Rights Reserved

C. Bench marking

Optimization of a cluster of R-Pi computers requires benchmarking at two different levels- 1) at the software and 2) the hardware configuration. The software deployed depends on the application and the range of use (scientific, Big Data, cloud computing, sensor-based applications et al). This suggests that multiple benchmarking would offer better insights to deploy those that best suit the application. (Hadoop, MPI/ Linpack, STREAM). The hardware considered for the R-Pi mainly for its efficient power utilization and because of its capability to handle sensor-data. The recent versions of R-Pi have multiple cores allowing for faster computing and improved data handling capabilities [18].

Two benchmarks that have found acceptance over others are the Linpack and Stream.

HPL (High performance Linpack) uses BLAS (Basic Linear Algebra subprograms) and the MPI (message passing interface) to rate the performance of machines under test. It is deployed to test supercomputers and is accepted as a standard protocol for performance of distributed networking structures and multicore and multiprocessor configurations [19].

STREAM tests the memory handling capabilities of a machine. Effectively, it reports the time taken for copying values by the software, adding the values, and scaling. [20].

The output of the HPL would be measured under the following heads: FLOPS (Floating Point Operations per Second): since the HPL is designed for much larger and faster CPUs; the number of operations chosen for the smaller R-Pi has to be smaller. Lesser number of cores and the networking interface may also contribute to the performance results.

FLOPS energy consumption is calculated in FLOPS/W. Newer versions of R-Pi that have dual core, quad core and more, and the latest processors that allow 64-bit handling instead of earlier 32-bit machines show much improved energy efficiency even when ARM processors are used. These efficiencies have reached the level of GFLOPS/W. The third important measure is the FLOPS/$ in which all versions of Raspberry Pi measure well because of very low procurement costs even though Intel processors seemingly pack more computing power [19].

D. Evaluation

In order to get a statistical evaluation of the measurements obtained from different tools, a scientific rigor suggests three distinct stages, namely: Data Preprocessing, Constructing a performance Space and finally Evaluation. Fig. 4 below illustrates the salient stages of establishing the benchmark evaluation methodology.

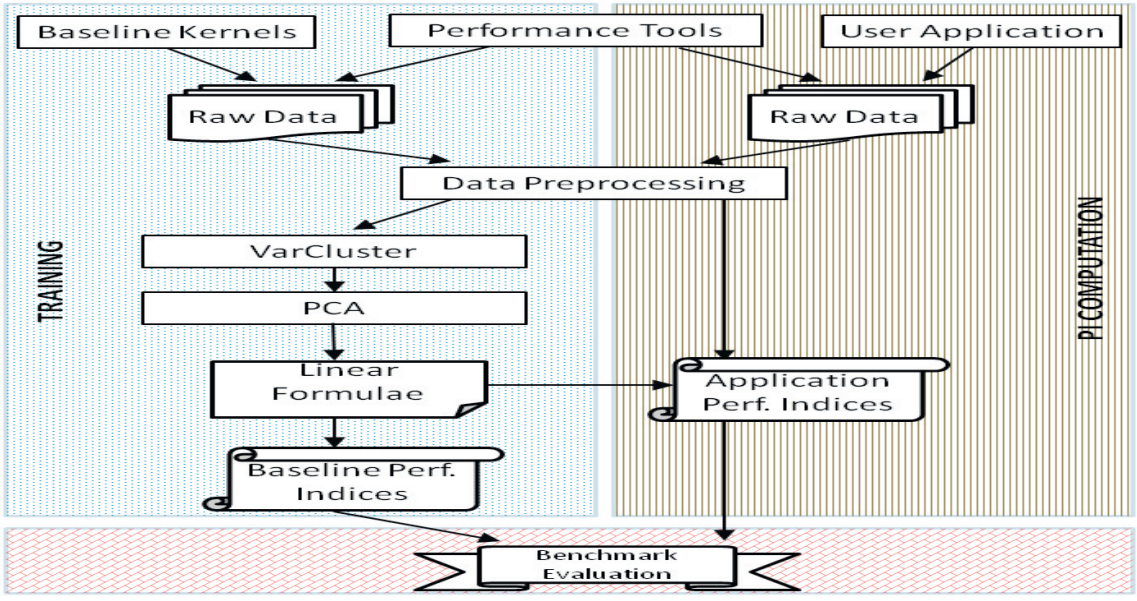

Fig. 4 Benchmark evaluation Flowchart [21]

The first step involves normalizing deriving each HPC per MPI benchmark to a comparable scale. The subsequent stage is to compute Performance Indices derived from a set of linear equations (accrued from statistical methods on the normalized HPC benchmarks).

© 2014- 17, IRJCS- All Rights Reserved

The evaluations obtained in the first step are then mapped on the Performance space created in the second step. This groundwork is then used for evaluation of performance indices of a cluster when applied to a particular application. Thus, the resulting matrix of evaluation would comprise of *n* x *m* members, where *n* is the number of HPC benchmarks and *m* is the number of Performance Indices [21].

Var Cluster is another important parameter of scientific evaluation process [7] that accounts for benchmarks with close correlation. The central theme lies in the homogeneity amongst variables in different clusters, thus accounting for the underlying commonality to simplify the task of deploying similar performance measures. Having thus brought all measurements under a common theme and those bearing homogeneity in different clusters, PCA (Principal Component Analysis) can be applied to obtain consistency in results [21].

E. Performance Tools

The rigorous performance tools accepted across platforms for HPC structures comprise of the HPC Challenge (HPCC), PAPI and FPMPI. The benchmark measures effective supercomputing parameters such the FLOPS, latency, communication bandwidth and memory access bandwidth. HPCC is easy to deploy as it employs common kernels: DGEMM, FFT, HPL, Latency Bdh, PTRANS, RA, and STREAM.

DGEMM measures a double-precision Dense Matrix Multiplication in real value.

FFT is the Fast Fourier Transform of complex value vector HPL solves linear algebraic equations using matrix multiplication through the established method of Gaussian Elimination.

Latency Bdh measures the communication capability of a cluster or even a multiprocessor through bandwidth of the communication channel in the multiprocessor or nodes in a cluster and the latency.

PTRANS (Parallel Matrix Transpose) is a measure the large data exchange communication of parallel data transfer between pairs of processors. RA is Random Access and pertains to the global memory space. The random assignment methodology requires to be assessed by either the embedded Finite field arithmetic logic or by the customized LCG (Linear Congruetional Generator).

STREAM measures the memory bandwidth vis-à-vis the rate of computation for vector operations thereof [21]. The advantage with HPCC is that it attends to Serial/Single processor, parallel processing that takes place in a standalone, independent mode (without communicating processes or results with other cores or processors) and MPI mode, in which processors are in communication with others and is essentially SMP or ASMP mode of operation. Finally, we come to the Performance tools that best suit the purposes of performance evaluation in a networking environment. In keeping with the aim of using the cost, ease of use and accessibility as the measure of choice, FPMPI and PAPI are found to be most suitable for profiling an R-Pi cluster. FPMPI (Fast Profiling MPI) [25] is lightweight library that measures and reports min/max/avg. for each metric it measures in all MPI processes. It helps understand the networking and communication pattern thereby exposing the bottlenecks and allowing the user to deal with such issues to improve the performance by restructuring the cluster network for efficient parallel processing. Additionally, the library calculates the load imbalance factor between MPI processes given by the ratio of difference between maximum and minimum values and the maximum value (max – min)/ max. A '0' indicates perfect load balance and '1' a heavy imbalance of tasks assigned to clusters. FPMPI is capable of assessing the level of balance or imbalance of load assigned to threads by calculating values between '0' and '1'.
PAPI (Performance Application Programming Interface) [22] library is the standard protocol for measuring the hardware performance. The API is cross-platform allowing processor performance interchangeably across tool development and application programming arenas. Specifically, when supported by architecture PAPI can measure computation capabilities by assessing the number of integer operations, floating –point arithmetic, memory access attempts and operational features such as CPU cycles required per operation and Branch prediction failures, and translation look aside buffer (TLB) and cache misses.

III. DISCUSSION

Amongst the parallel programming architectures the three most commonly used are openMP, MPI, and MapReduce framework. Each has its own features to suit different data handling capabilities. OpenMP is preferred when shared memory systems are required. For distributed memory systems, MPI is the preferred choice whereas when the tasks involve dat-intensive handling as in iterative rigor, Map Reduce becomes the chosen industry standard. In order to arrive at the right choice amongst the three, benchmark values from typically two types of problems seem to offer the best criterion: 'data-join problem' and the 'all-pairs-shortest-path-problem'.

© 2014- 17, IRJCS- All Rights Reserved

Since the evolving versions of R-Pi offer multicore technology, a Beowulf cluster built using many such small processors act as threads where a large, heavyweight task can be approached by breaking it down into assigned smaller lightweight tasks that run concurrently. The openMP [23] and POSIX thread library [24]) is a typical implementation of this strategy. In this architecture, memory is shared and avoids the need for constant messaging for access. Where memory is not shared and processors depend on localized memory inventories, the execution requires constant message exchange between threads and nodes, and this need is best served in the MPI deployment [25]. Effectively, coordination of multiple nodes in the threads of task segments helps complete heavy weight tasks, though with the unavoidable disadvantage of latency issues. However, improved versions of faster memory architectures (DDR2, DDR3 and DDR4) obviate the issues to a large extent.

Hadoop is typically used when the data volumes are very large and memory manipulation and processing becomes prohibitive for the above frameworks. In such cases as Big Data, Hadoop breaks down the application into two phases: Map and then Reduce, making it easier to assign and track the progress of data computation [26]. An improvement over MapReduce programming model is the PACT (Parrallelization Contracts). Pact allows programmers to specify runtime allocations while allowing processes that need access to multiple inputs. Effectively, PACT can handle more complex tasks, accommodate specific command controls and work in a more complex environment (as required in graph algorithms, data mining or relational queries), with more inputs [26]. The additional features PACT programming model adds to MapReduce are the additional Input Contracts. Three of those are : a) Cross: creates a Cartesian product on multiple key inputs; b) CoGroup: creates a subset of keys with same values, and c) Match: works on multiple inputs and matches those in the same subset, processing them independently [26]. openMP creates a set of threads, synchronizes operations between them and allows for efficient use of common memory space through compiler directives that it generates on top of pThreads. Effectively, for the programmers, it is easier to use as the API (application programme interface transforms the sequential tasks into parallel multithreaded programs. [27].The programmers can use openMP without a serious understanding of multithreaded mechanism. The API creates the runtime to maintain threaded pool for which it deploys sets of libraries [2]. The underlying design layout is a block-structure Strategy. Here, the progress is that of an alternate progression of sequential and parallel blocks. At the start of a parallel string, the tasks are split into threads after finishing which; a new sequential thread is treated similarly. The main advantage of this framework is the total control over threads and that it can run over Linux, Unix, and Windows – the most widely used platforms for all types of applications and is supported by high level languages; FORTRAN, C, and C++ [27].

MPI is essentially a tool for passing messages in a parallel, distributed computing structure. The programmers have set directives for specified progress mechanism sought for the execution of tasks in a parallel environment. The memory addresses are allocated to each process in different clusters and nodes and message passing makes it possible to access other memory spaces. The partitioning of tasks and allocation has to be selected and assigned according to the sequential requirements. This strategy then can be summed up as a set of node-oriented, collective broadcasting, and parallel I/O communication through effective message passing [25]. MPI can be implemented by users on platforms such as Windows, Linux and Solaris. MPI implementations use NFS (network file systems) used by Hadoop. Thereby they can be used on single multiprocessors (with multicores) or even by a multimode clusters as in Beowulf under Hadoop. MPI allows programmers to create efficient threading for larger tasks in consolidation with the multiprocessing, parallel as well as concurrent requirements [28].

Hadoop uses a proprietary MapReduce strategy that creates a distributed File Structure (HDFS) to accommodate and facilitate processing of huge volumes of data [29]. The methodology involves mapping and then reducing tasks as pairs. In the initial stage, the input data is partitioned and the keys generated thereby are reduced to task assignments to cluster segments. This methodology absolves the programmers the rigor of assigning threads and assigning properly segmented tasks (that may then run the risk of overlap or even repetition).Thus, the MapReduce technique deployed by Hadoop is an efficient and faster programming facility when used in tandem with MPI. MapReduce, additionally absolves the need to load the entire data, as threads can execute jobs without the need for essential partitioning, as each segment works independent of the other. Thus, MapReduce is the preferred choice when Big Data analytics and computation is required. Further, PACT can replace Hadoop to suit more complex and multiple input implementations.

IV. CONCLUSIONS

Even as we realize the advantages openMP offers over MPI, we should also be cognizant of the limitations of the MapReduce architecture. For example, MapReduce works only with simpler applications with limited inputs. An improvement over this shortcoming is the PACT (Parallelization Contracts).

© 2014- 17, IRJCS- All Rights Reserved

Thus, the choice of architecture for improving cluster performance depends on the vastness of computation, complexity and specifications that programmers need to embed. Evolving software technologies and increasingly powerful Raspberry Pi versions are making clustered formations efficient enough to replace costlier PC nodes making them viable alternatives for researchers and students at the laboratory level; especially because of the portability and economic considerations.

REFERENCES

1. D. Culler, J. P. Singh, and A. Gupta, Parallel Computer Architecture: A Hardware/Software Approach, 1st edition. San Francisco: Morgan Kaufmann, 1998.
2. J. Diaz, C. Muñoz-Caro, and A. Niño, "A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era," IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 8, pp. 1369–1386, Aug. 2012.
3. L. K. Hyung, K. A. Seong, and K. M. Lee, "Hierarchical partition of nonstructured concurrent systems," IEEE Trans. Syst. Man Cybern. Part B Cybern. Publ. IEEE Syst. Man Cybern. Soc., vol. 27, no. 1, pp. 105–108, 1997.
4. K. M. Lee and K. M. Lee, "Similar pair identification using locality-sensitive hashing technique," in The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems, 2012, pp. 2117–2119.
5. S. W. Lee et al., "Real-Time System-on-a-Chip Architecture for Rule-Based Context-Aware Computing," in Knowledge-Based Intelligent Information and Engineering Systems, 2005, pp. 1014–1020.
6. A. C. Sodan, "Message-passing and shared-data programming models - wish vs. reality," in 19th International Symposium on High Performance Computing Systems and Applications (HPCS'05), 2005, pp. 131–139.
7. R. Rabenseifner and G. Wellein, "Comparison of Parallel Programming Models on Clusters of SMP Nodes," in Modeling, Simulation and Optimization of Complex Processes, Springer, Berlin, Heidelberg, 2005, pp. 409–425.
8. M. F. Cloutier, C. Paradis, and V. M. Weaver, "A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement," Electronics, vol. 5, no. 4, p. 61, Sep. 2016.
9. H. Karchalkar and P. Railkar, "Raspberry Pi Hadoop Cluster based Data Processing," IJCA Proc. Int. Conf. Internet Things Gener. Netw. Cloud Comput., vol. ICINC 2016, no. 2, pp. 1–3, Jul. 2016.
10. Lopez, "Hadoop on a Raspberry Pi," Datanami, 27-Nov-2013. [Online]. Available: https://www.datanami.com/2013/11/27/hadoop_on_a_raspberry_pi/. [Accessed: 16-Mar-2017].
11. I.-Y. Jung, K.-H. Kim, B.-J. Han, and C.-S. Jeong, "Hadoop-Based Distributed Sensor Node Management System," Int. J. Distrib. Sens. Netw., vol. 10, no. 3, p. 601868, Mar. 2014.
12. A. Holmes, Hadoop in practice. Shelter Island, NY: Manning, 2012.
13. S. Franks and J. Yerby, "Creating a Low-cost Supercomputer with Raspberry PI," in Proceedings of the Southern Association for Information Systems Conference, 2014.
14. S. Vijaykumar, M. Balamurugan, and K. Ranjani, "Big Data: Hadoop Cluster Deployment on ARM Architecture," in ResearchGate, 2015, vol. 4, pp. 56–59.
15. M. Maurya and S. Mahajan, "Performance analysis of MapReduce programs on Hadoop cluster," in 2012 World Congress on Information and Communication Technologies, 2012, pp. 505–510.
16. M. Sekiyama, B. K. Kim, S. Irie, and T. Tanikawa, "Sensor data processing based on the data log system using the portable IoT device and RT-Middleware," in 2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2015, pp. 46–48.
17. V. Govindaraj, "Parallel Programming in Raspberry Pi Cluster," Cornell University, 2016.
18. M. F. Cloutier, C. Paradis, and V. M. Weaver, "A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement," Electronics, vol. 5, no. 4, p. 61, Sep. 2016.
19. M. Geveler, M. Köhler, J. Saak, G. Truschkewitz, P. Benner, and S. Turek, "Future Data Centers for Energy-Efficient Large Scale Numerical Simulations," in Proceedings of the 7th KoMSO Challenege Workshop: Mathematical Modeling, Simulation and Optimization for Energy Conservation, Heidelberg, Germany, 2015, pp. 8–9.
20. A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, "HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers," netlib.org, 24-Feb-2016. [Online]. Available: http://www.netlib.org/benchmark/hpl/. [Accessed: 16-Mar-2017].
21. F. Xing, H. You, and C. Lu, "HPC Benchmark Assessment with Statistical Analysis," Procedia Comput. Sci., vol. 29, pp. 210–219, Jan. 2014.
22. B. D. Garner, S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A Portable Programming Interface for Performance Evaluation on Modern Processors," Int. J. High Perform. Comput. Appl., vol. 14, pp. 189–204, 2000.
23. "IEEE SA - POSIX - Austin Joint Working Group." [Online]. Available: http://standards.ieee.org/develop/wg/POSIX.html. [Accessed: 22-Mar-2017]."spec30.pdf.".

© 2014- 17, IRJCS- All Rights Reserved

24. W. Gropp et al., MPI - The Complete Reference, 1998 ed., vol. 2. London: MIT PRESS, 1998.
25. A. Alexandrov, S. Ewen, and M. Heimel, "MapReduce and PACT - comparing data parallel programming models," in BTW 2011. Datenbanksysteme für Business, Technologie und Web. CD-ROM, 2011, pp. 25–44.
26. B.Barney, "Introduction to Parallel Computing," 19-Jun-2016. [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/. [Accessed: 22-Mar-2017].
27. G. Jost, H.-Q. Jin, D. anMey, and F. F. Hatay, "Comparing the OpenMP, MPI, and hybrid programming paradigm on an SMP cluster," 2003.
28. X. Tian and B. R. De Supins, "Explicit Vector Programming with OpenMP 4.0 SIMD Extension," Primeur Mag., vol. 2014, 2014.

© 2014- 17, IRJCS- All Rights Reserved