FEPAC: A Framework for Evaluating Parallel Algorithms on Cluster Architectures

Mehul Vikas Warade, Jean-Guy Schneider and Kevin Lee
School of Information Technology
Deakin University, Geelong
Geelong, Victoria
{mwarade,jeanguy.schneider,kevin.lee}@deakin.edu.au

ABSTRACT

For many years, computer scientists have explored the computing power of so-called computing clusters to address performance requirements of computationally intensive tasks. Historically, computing clusters have been optimized with run-time performance in mind, but increasingly energy consumption has emerged as a second dimension that needs to be considered when optimizing cluster configurations. However, there is a lack of generally available tool support to experiment with cluster and algorithm configurations in order to identify "sweet-spots" with regards to both, run-time performance and energy consumption, respectively. In this work, we are introducing FEPAC, a framework for the automated evaluation of parallel algorithms on different cluster architectures and different deployments of software processes to hardware nodes, allowing users to explore the impact of different configurations on run-time properties of their computations. As proof of concept, the utility of the framework is demonstrated on a custom-built Raspberry Pi 3B+ cluster using different types of parallel algorithms as benchmarks.

KEYWORDS

Cluster Computing, Evaluation Framework, Energy-Aware, Parallel Algorithms, Single Board Computers

ACM Reference Format:
Mehul Vikas Warade, Jean-Guy Schneider and Kevin Lee. 2020. FEPAC: A Framework for Evaluating Parallel Algorithms on Cluster Architectures. In *Proceedings of Australasian Symposium on Parallel and Distributed Computing (AusPDC'21).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Data driven technologies are demanding increasingly complex analysis and management to produce effective results. As predicted by Moore's law [19], advances in processors and technology has led to increased performance of modern hardware. Algorithms are used to reduce the manual work and effectively utilise the ever-growing computer technologies to their full extent. Newer technologies and advancements led to exponential growth in the amount of data that needed processing [3].

Parallel processing allows computation to be performed faster by utilising a number of processors concurrently [10]. Parallel computing provides us with the technology and means to achieve the ever-increasing demands of computing. Clusters are designed with performance in mind which has led to the creation of huge data centres that have heightened the energy demand [12]. Energy consumption is one of the top challenges for the next generation of super-computing [11], but it is difficult for a researcher to determine the optimal configuration of cluster in terms of performance and energy, respectively. A challenge in the field of parallel computing is the lack of an evaluation method which focuses on energy consumption and the factors influencing it.

Clusters are expensive and difficult to experiment with and thus it remains difficult to optimise for energy-efficiency. Single board computers (SBCs) are complete computers developed on a miniaturised single circuit board. Low cost clusters of SBCs can help solve this challenge by allowing experimentation and wide range of configurations. They also have the ability to perform reasonably well when compared to contemporary devices in terms of cost and power consumption [7]. Due to their multi-core nature, they can be optimised to achieve computation faster by using their inherent parallel capabilities.

The aim of this work is to support researchers in investigating the factors affecting the performance and energy consumption of parallel computations. It does this by proposing a framework which evaluates the execution of parallel algorithms on varying cluster configurations with a focus on performance and energy consumption of the computation. It will allow researchers who use parallel computation to test the performance of their algorithm in a variety of cluster configurations in order to determine the best configuration for their needs. The framework is evaluated with three distinct parallel algorithms and a cluster composed of Raspberry Pi 3B+ SBCs which have been proven to be useful proxies for large-scale cluster computing [28]. The framework will likely lead to cost reduction of computations as well as empowering research in energy efficient computation methods.

The remainder of this paper is organized as follows. Related work is reviewed in Section 2. Section 3 introduces a framework to support evaluation of parallel algorithms on different cluster architectures. Section 4 presents an experimental evaluation of the framework using a cluster of SBCs. Section 5 provides a discussion of the experimental evaluation. Finally, Section 6 summarizes key observations and describes planned future work.

AusPDC'21, 1-5 February 2021, Australia/New Zealand
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Computing power available for researchers have been increasing exponentially [27]. Parallel computing have been used to harness the computational power from individual computing units using clusters. As well as maximising performance, there is a need to design clusters which can perform under a given power budget [7]. Clusters consume a large amount of electrical power and require sufficient cooling for optimal performance. ($GFLOPS/W$) is used to measure the power and energy consumption of clusters whereas ($GFLOPS/\$$) is used measure the effectiveness of the cluster in terms of performance and cost. This section reviews literature in experimental cluster computing, energy aware clusters, and benchmarking clusters.

2.1 Experimental Cluster Computing

Cluster computing is used heavily in scientific computation and big data processing. Scientific workflows are executed on cluster computing infrastructure using workflow engines [15, 17]. Using clusters for computation requires solutions to issues such as data storage, data retrieval and file handling for cluster computing. Apache Hadoop [30] aims to solve this problem through technologies such as the Hadoop Distributed File System (HDFS) [4]. Generally, full clusters such as those using Apache Hadoop are very expensive and requires a lot of energy, real estate and custom cooling. SBC clusters have been proposed as an effective alternative for mobile Hadoop clusters and robust computing performances [25, 29, 30].

Novel cluster architectures were introduced to tackle the problems of Edge computing and Big Data. PiStack [2] focuses on power efficiency and thermal output while providing an optimal performance in edge computing conditions. It reduces hardware footprint by powering nodes through the cluster case and saving power by introducing heartbeat functionalities for each node. Deployment approaches include 1U rack mounting of SBC clusters in data centres to achieve maximum accessibility and easier replacement [29]. This demonstrates that SBC clusters can be a feasible approach to edge computing clusters for on-site big data processing.

Using SBC clusters to perform cloud simulations and provide virtualisation of resources have been proposed by [13, 32]. Cloud infrastructure has been simulated using clusters in iCanCloud [20] and CloudSim [6]. The Glasgow Raspberry PiCloud [32] has used SBCs to create a cluster for simulation processing. PiCloud [32] simulates every layer of cloud computing infrastructure.

SBC clusters are particularly useful for edge computing such as those performing on-site image processing. Image processing is one of the applications which can make use of in-built parallelism during processing. Algorithms in OpenCV [30] libraries exploit parallelism to process each image faster. Clusters executing algorithms which use the OpenCV library significantly outperformed single computers in frame processing of a live stream video [24]. The accuracy of SBC clusters in image learning was studied by using the Scikit Image library and by executing two parallel algorithms - Watershed and Edge detection on number of images [18]. A comparative study of performance of different image processing libraries using OpenMP [28] on clusters is presented in [23, 26].

2.2 Energy Aware Cluster Computing

The Apache Hadoop [30] framework is commonly used for analysis of data intensive operations such as Big Data analysis where large volumes of data need to be analysed effectively [25]. Hadoop's Map/Reduce model is a useful benchmarking tool for comparing performance and energy consumption of clusters [9]. Comparative studies of energy consumption in Hadoop clusters show that SBC clusters can be an effective alternative [8, 14, 25, 31].

Data mining algorithms are used to extract information from big data-sets [28]. [28] compared two data-mining algorithms (Apriori and K-means) on a SBC cluster and an HPC platform. They concluded that even though SBC cluster provided lower performance than HPC platform, they can be an effective energy-efficient alternate. Cloud computing is used to process a large number of computations remotely in data centres [25]. Due to energy costs, $GFLOPS/W$ is an important consideration for cloud computing infrastructure [2]. A predictive optimisation model for balance between performance and energy consumption in cloud computing is presented in [5]. SBC clusters have been compared to find the best architecture to provide maximum performance in terms of low network latency, communication overhead, low power and energy consumption [2, 25].

2.3 Benchmarking Cluster Computing

Evaluating and benchmarking a cluster provides a value of its maximum performance. There are many benchmarking libraries and frameworks developed to test different aspects of a cluster. [22] used High Performance Linpack (HPL) as a benchmark to test the performance of high performance Beowulf cluster comprising of 12 node Raspberry Pi 2B SBC. [1] benchmarked SBC clusters comprising of Pandaboard ES boards and Raspberry Pi boards using 4 different benchmarks (CoreMark [16], STREAM [16], Linpack [16], HPL [16], Ping Pong [16] and Nas Parallel [16]). Benchmarking can also reveal characteristics other than raw performance; [1] concluded that raw performance of Pandaboard ES was 2.5 times greater than Raspberry Pi SBCs but with closely related in terms of performance per watt.

To support researchers, benchmarks have been created for workloads. [33] developed a framework to benchmark two state of the art workloads in assessing elasticity in graph analytics. The framework helps in benchmarking and understanding the benefits, costs and resource management efficiency of workloads. [34] developed the BigOP framework to benchmark workloads on big data systems (Hadoop and Spark). [21] conducts proxy benchmarks on databases used in big data systems (MySQL, Cassandra, MongoDB). These evaluate the performance of workloads and provide comparative results.

The work discussed here shows that cluster computing is mainly focused on implementing parallel and cluster computing to solve specific problems but not on improving the efficiency of already implemented systems. This is mainly due to lack of models that identify and address the factors that can help in improvement of the efficiency. Our approach fills that gap by proposing a generic framework to evaluate different algorithms on varying cluster configurations while focusing on their effects on the energy consumption of the cluster.

3 FRAMEWORK DESIGN

In this section, we describe a generic high-level design of the "Framework for Evaluation of Parallel Algorithms on Clusters" (FEPAC), a light-weight and flexible framework for the evaluation of parallel algorithms on different cluster configurations. The framework must fulfill several requirements that can be derived from its objective as a suite to evaluate the performance of parallel programming models. These criteria will function as guidelines for the selection of benchmarks as welll as benchmarking practice.

3.1 Framework Requirements

The design of the framework was guided by the following requirements:

- R1 The framework shall help researchers in comparing run-time performance with energy consumption.
- R2 The framework shall support the identification of bottlenecks for a given computation.
- R3 The framework shall allow the specification of a range of repeated and repeatable experiments with varying algorithm parameters and cluster configurations.
- R4 The framework shall automatically configure the target cluster for the computation needs as defined in the experiment's configuration.
- R5 The framework shall automatically execute different algorithms on different cluster configurations.
- R6 The framework shall support algorithms from different programming languages.
- R7 The framework shall provide a comparative output (within suitable time) of energy consumption for different cluster configurations and algorithms.
- R8 The framework shall support exporting experimental data for further analysis.

Designing a parallel computing system is a multifaceted process. It involves individual designing of hardware and software components. Following the requirements provided above, the design of the proposed framework is given below.

3.2 FEPAC: A Framework for Evaluating Parallel Algorithms on Cluster Architectures

Figure 1 illustrates a generic cluster architecture as used in the design of the framework presented here. It comprises a number of inter-connected computing nodes. The master-slave model used in the design includes asymmetric communication in which a device (master) controls one or more other devices (slaves or nodes). Each node has its own operating system, memory and power source.

A high-level view of the framework is illustrated in Figure 2. Based on the requirements outlined in Section 3.1, the functionality of the framework can be identified and implemented accordingly to full-fill the requirements completely. The functionality that is important and is compulsorily needed to be

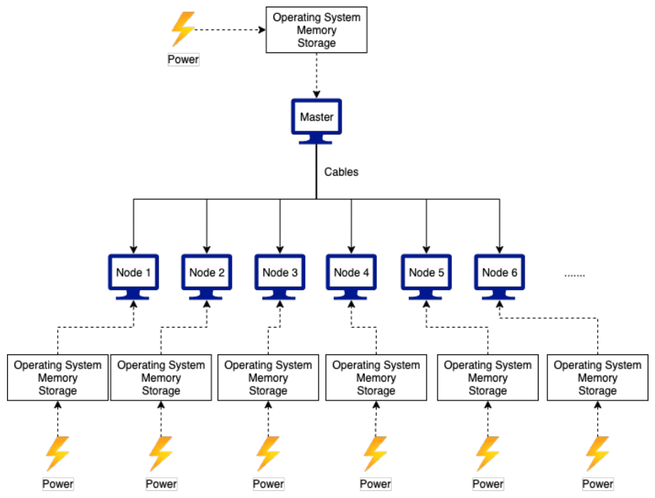

Figure 1: Generic Cluster

implemented is included in the generic design of the framework. The framework provides researchers with a configuration file that includes all the information needed to execute computations on a variety of cluster configurations. The functionality implemented in the framework is outlined below.

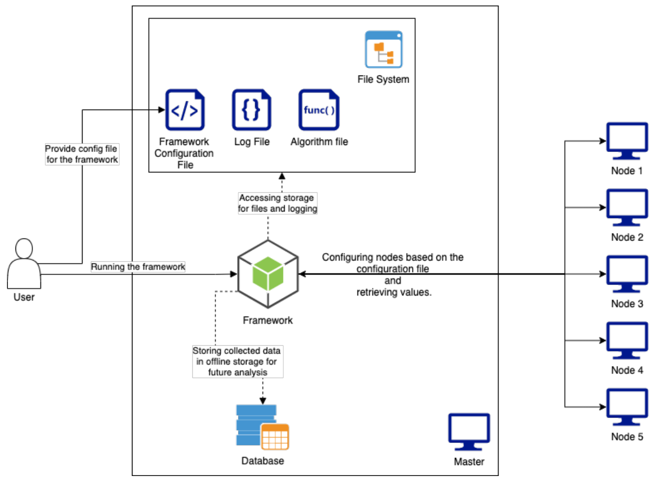
File System
Framework Configuration File
Log File
Algorithm file
Provide config file for the framework
Accessing storage for files and logging
User
Running the framework
Configuring nodes based on the configuration file and retrieving values.
Framework
Storing collected data in offline storage for future analysis
Database
Master
Node 1
Node 2
Node 3
Node 4
Node 5

Figure 2: Proposed Design of the Framework

Cluster Setup. The framework can automatically set up the cluster and its nodes for computation (R4). This can be achieved by using the MPI protocols during run-time (refer to Figure 3). The user provides the framework with the preferred configuration using the configuration file. Once executed, the framework will iterate through the framework as configured, send computation to the cluster nodes and collect data.

Problem Splitting. The framework provides each node with their individual slice of data to compute on (R5 and R6). This can also be achieved through MPI protocols. The data on the master node

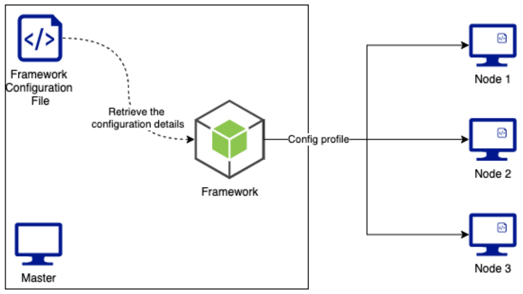
Framework Configuration File
Retrieve the configuration details
Config profile
Framework
Master
Node 1
Node 2
Node 3

Figure 3: Proposed set-up of nodes in the cluster

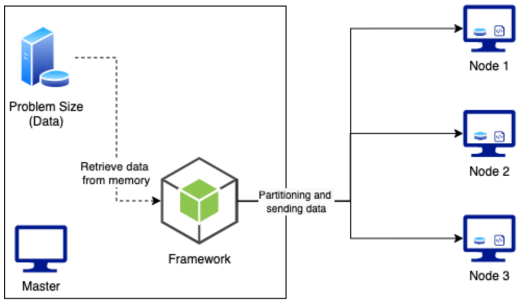

Figure 4: Splitting of data for cluster computing

is sliced into smaller chunks so that each node has a small part to compute and return the results to master. This is the main essence of parallel computing – to reduce the computation time and increase performance by reducing the work load on individual nodes by distributing the overall computing work across number of nodes.

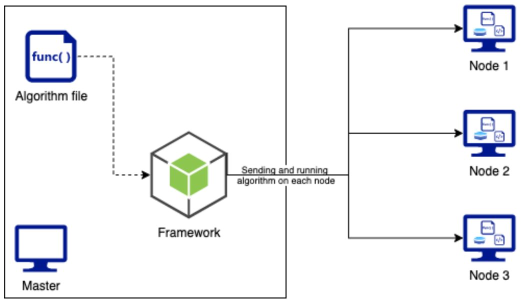

Figure 5: Execution of an algorithm in a cluster

Algorithm Execution. The framework needs to provide the nodes with the actual set of instructions to perform the required computation (R5 and R6). The user needs to provide the algorithm file and declare its usage in the configuration file. The configuration can include specifications such as location of the file, parameters needed for algorithm, the dependencies, the output format, and other needed factors which are important in executing the experiments. The framework communicates the algorithm file to each node and the nodes execute the algorithm as per the specifications provided by the framework (refer to Figure 5).

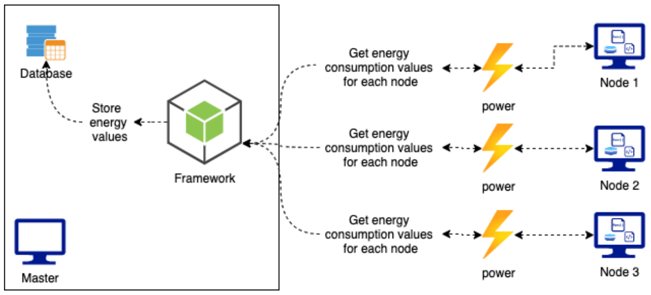
Database
Store energy values
Framework
Master
Get energy consumption values for each node
power
Node 1
Get energy consumption values for each node
power
Node 2
Get energy consumption values for each node
power
Node 3

Figure 6: Collecting energy consumption values

Data collection. R7 can be achieved through implementing functionality to monitor and log the energy consumption of each node. Measuring the energy consumption of a particular node is difficult to achieve using purely software methods. For accurate measurements there needs to be a hardware interface installed at the power source which can monitor and send the energy consumption data to the framework. This can be achieved through a number of steps. A digital power source with sensors can help in collection of the energy data (see Figure 6).

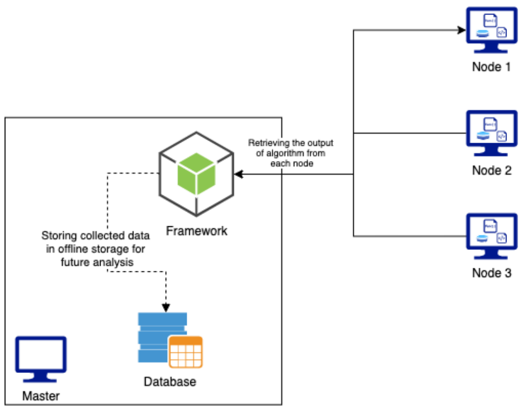
Retrieving the output of algorithm from each node
Framework
Storing collected data in offline storage for future analysis
Database
Master
Node 1
Node 2
Node 3

Figure 7: Collecting and storing algorithm output data

Data storage. Collecting data from program execution is an important aspect of any framework. Algorithms can have many different outputs ranging from log entries to results of the computation. The framework cannot and should not restrict users on the output their algorithm provides and hence, a framework should be able to accommodate whatever output the algorithms generates. This functionality will help in achieving R3, R8 and R1. The data collected through this medium is the raw data generated by a particular algorithm and can help the researcher in monitoring, evaluating or debugging the algorithm when needed.

Data Exporting. Data exporting is needed for further processing of experimental results and/or archiving these. Exported data may also lead to future improvements by implementing new functionality to use this data. The framework exports the data from computations into a local database. Users can then export

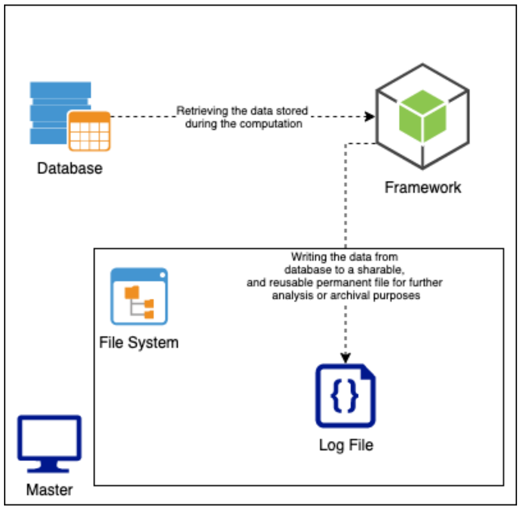
Retrieving the data stored during the computation
Database
Framework
Writing the data from database to a sharable, and reusable permanent file for further analysis or archival purposes
File System
Log File
Master

Figure 8: Exporting collected data to a file

that data into a compatible file format which can be used to analyse the data further or add new functionality to the framework. This functionality allows researchers more control over the data and freedom to analyse the data in number of ways. R8 can be full-filled through this design and this can help in achieving R2.

4 EXPERIMENTAL EVALUATION

FEPAC is a platform-independent framework that allows a parallelisable algorithm to be executed repeatedly on increasing sizes of cluster. It allows the collection of experiment data including energy consumption (both the cluster and individual nodes), run-time and algorithm output. Raw data can be exported to be used for further analysis of the experiments. This section presents an experimental evaluation of the framework as evidence of how it achieves the goals of this paper. The following experiments attempt to cover a range of scenarios by using 3 distinct algorithms, used in a wide range of domains. Table 1 briefly summarises the algorithms evaluated and their parameters.

Name	Computation Details
Matrix Multiplication	Dot product of 3000x3000 and 240x240 matrices
k-means	Clustering of 288000 data points into 48 clusters
OpenCV Filtering	5 Images (1920x1080 pixels) Applied filters on each image: blur, sepia, emboss, warm, cold and increased brightness.

Table 1: List of Algorithms Evaluated and their Parameters

4.1 Experiment Setup

The implementation of the cluster for this experimental evaluation involved eight individual Raspberry Pi 3B+ SBCs connected to a managed switch as illustrated in Figure 1. Seven of the eight RPi3B+ are computing nodes while one RPi3B+ acts as a networking configuration node which provides the required network configurations necessary to manage the cluster.

Operating System. For the purposes of this experimental evaluation, the operating system used on the cluster nodes will mainly facilitate the use of the platform for computation. To minimise overhead, a lightweight and highly optimised Operating System is needed to allow most resources for computation. DietPi, a Debian-based Linux distribution was chosen, as it is primarily developed for single board computers and highly optimised to enable maximum performance.

Networking. RPi's are usually bootstrapped by flashing an Operating System image onto a SD card and then inserting the SD card into the RPi. Due to the light-weight nature of DietPi, each node can instead be network booted which requires no local drive for easier experiment configuration. A server node handles all network booting and provides a base file system for enabling proper functioning of the nodes using NFS.

Energy Monitoring. To power all the nodes in the cluster, Power-over-Ethernet (POE) from the managed switch was used. Energy consumption was monitored by performing queries to the switch via a Telnet connection. Energy readings provided by the switch give the instantaneous power consumption in Watts for each network power, therefore giving the power consumption for all the nodes in the cluster. With repeated experimentation and measurement, these values have proved to be reliable and consistent. This data was retrieved as a string, sliced, and then stored in a MySQL database.

Physical Setup. The experimental cluster setup can be seen in Figure 9. Six RPi3B+ were mounted on top of each other. Two RPi3B+ were mounted together to distinguish them from the cluster. The top RPi is the one providing the network boot and file-system to other nodes (Denoted by B) whereas the bottom one is the master node (Denoted by M). The six RPiś were purely used for computing purposes (labelled as S1-S6).

Figure 10 shows an extract of the output file generated by the framework. It is exported in JSON format for easier use and cross-compatibility. The first object identifies the name of the algorithm. The second and third level objects represent the number of nodes and threads the algorithm is being executed on, respectively. The fourth level object is a tuple data structure comprising of the timestamps of that particular experiment and the average power consumption during the experiment.

4.2 Stability and Validity of Experimental Setup

To validate the experimental setup and verify the stability of running experiments, an evaluation of the framework by repeatedly executing the function np.dot for matrix multiplication on matrices with 240 x 240 elements was performed. The experiment required a number of basic Python libraries to be installed: (i) numpy, a library supporting computation with large arrays, and (ii) mpi4py, a Python binding for the Message Passing Interface (MPI) standard. The matrix multiplication was repeated

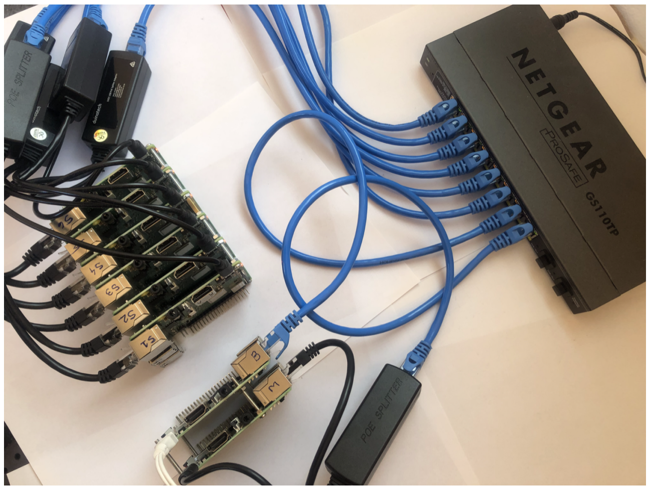

Figure 9: Photo of the Project Cluster Setup

```
"opencv": {
  "1": {
    "2": {
      "timestamp": {
        "start": "1598965940040",
        "end": "1598966009730"
      },
      "pwr_avg": {
        "p1": 4.239130403684533
      }
    }
  },
  "6": {
    "3": {
      "timestamp": {
        "start": "1598966016296",
        "end": "1598966149690"
      },
      "pwr_avg": {
        "p1": 4.513533785827178,
        "p2": 3.0887217467888854,
        "p3": 3.0127819534531213,
        "p4": 3.0721804253140785,
        "p5": 3.0338345864661656,
        "p6": 3.0563909344207074,
      }
    }
  }
}
```

Figure 10: Exported data in a JSON format

10 times for each configuration: 1 to 6 nodes (unused nodes were powered off) with each node executing 1 to 12 threads. The first thread on node #1 was configured as master thread and hence did not execute any matrix computations.

Figures 11 and 12 illustrate the distribution of the numerical energy consumption and run-time data of the matrix multiplication obtained from the repeated experiments. The x-axis represents the node configurations, varying from 1 node and 2 threads to 6 nodes and 12 threads, respectively. The two box plots illustrate the stability of the data collected from the experimental set-up used – the largest variation is from using 1 thread on each node as well as running the computation on a single node. The results illustrate the stability of the data collection and allow us to report experiments based on the average of the data from repeated experiments.

4.3 Algorithm Evaluation: Matrix Multiplication

Matrix multiplication is a common operation used in a wide range of fields, from mechanics in physics, graph theory in mathematics to gene expression in biology. Matrix multiplication was chosen as it has been heavily optimised to work with parallel computers. The experimental setup is no different from the one in previous section. The framework is provided with a parallel implementation of a matrix multiplication and parameters in order to execute the function np.dot for matrix multiplication on matrices with $3,000$ x $3,000$ elements.

Figures 13 describes the energy consumption of the algorithm. Matrix multiplication produces the results expected from a parallel computing algorithm. Increase in the number of computing nodes leads to decrease in the computing time and increase in performance. It was seen that the time to compute slowly starts flattening out at the end of 6 nodes. The power consumption is expected to go up as more nodes are added but in this case, the decrease in power consumption can be explained by the speedup obtained during the computation. As seen in Figure 14, the computation time of the algorithm reduces considerably when more computation power is provided. For this reason, even though energy consumption of the cluster increases due to increase in a computing node, the resulting speedup can help compensate for it.

Figure 15 shows the energy and the run-time performance data collected by the framework executing matrix multiplications. A double axis graph is shown for easy co-relation of both data-sets. The left Y-axis represents energy consumption values of the whole cluster and the right hand Y-axis represents run-time.

A researcher can calculate theoretically or experimentally the lowest computation time achievable for the give data set and algorithm using the results. This is very useful in predictive analysis where the algorithm output can be predicted based on its past performance. Researchers can also identify the bottlenecks from the graph saying that addition of each node degrades the performance and then using the nodes to their maximum capacity leads to increase in performance. The bottleneck here might be the cache memory which has to load the array first time any node is added to the cluster.

4.4 Algorithm Evaluation: OpenCV

OpenCV is a development library of aimed at real-time computer vision [18]. It is mainly used in image and video processing where a large amount of data needs to be processed in real time. The main aim of the experiment is to find the effects of image processing libraries and its different methods on the energy consumption of a cluster and to identify its bottlenecks. As shown in Table 1 six different types of filtering methods were used on 5

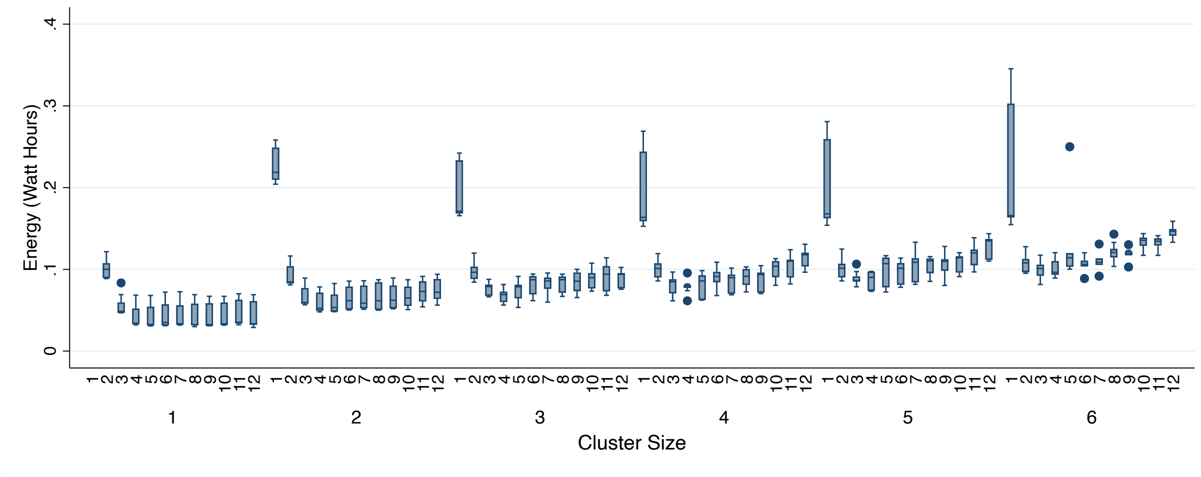

Figure 11: Energy Consumption for Matrix Multiplication 240 x 240 matrix (10 repetitions)

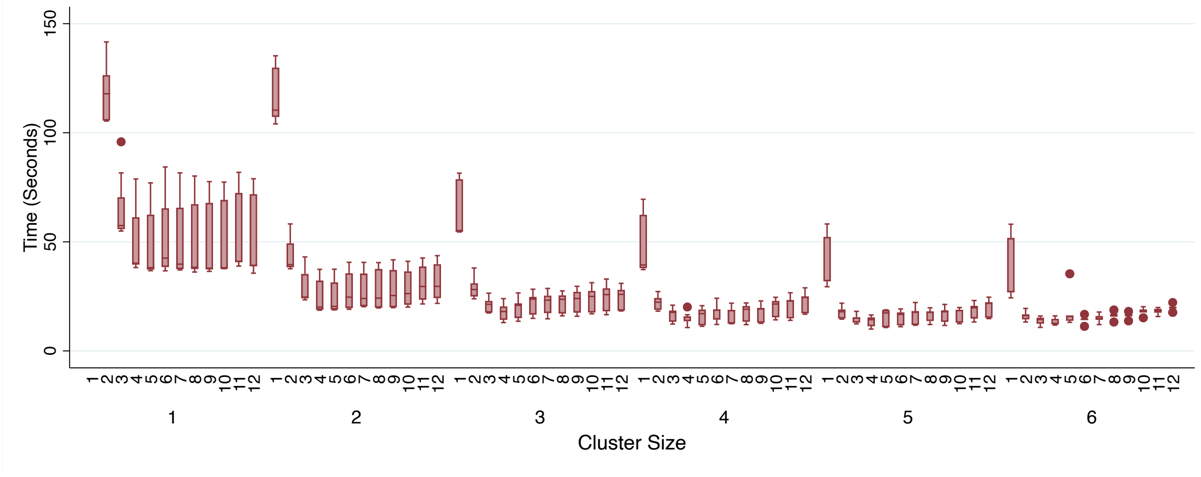

Figure 12: Time Consumption for Matrix Multiplication 240 x 240 matrix (10 repetitions)

high-res images recursively. The algorithm was developed in Python language. Experimental setup was similar to that of matrix multiplication with different supporting libraries installed (opencv (3.2.0+dfsg-6)).

Figure 16 illustrates the results of using the FEPAC framework to evaluate the OpenCV algorithm on increasing sizes of cluster. It illustrates the Energy Consumption of the cluster and the time taken by the algorithm to finish the computation, respectively. Figure 16 shows the energy and the run-time performance data collected by the framework executing OpenCV algorithm. The left Y-axis consists of energy consumption values of the whole cluster and the right hand axis comprises of run-time. Using the graph a clear crossover point between the energy and run-time can be concluded.

As expected, the energy consumption increases with an increase in nodes and resources being used. There is no significant trend in the time required by the algorithm, but a clear bottom limit of the speed can be seen, after which the algorithm starts to degrade performance. The framework executed the algorithm with the given configuration and provided the results shown in the graphs. From the graphs bottleneck for the algorithm can be hypothesized - data set being too small for computation. No clear trend in the data can also suggest researchers to find better ways to optimise the algorithm or motivate them to find the reason behind it.

4.5 Algorithm Evaluation: k-means

The k-means algorithm is a method that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean [28]. This algorithm was chosen as the algorithm heavily relies on the communication between master and slave nodes for computing. k-means have less independent work units which leads to more synchronization when the parallel work finishes. A C implementation of k-means is

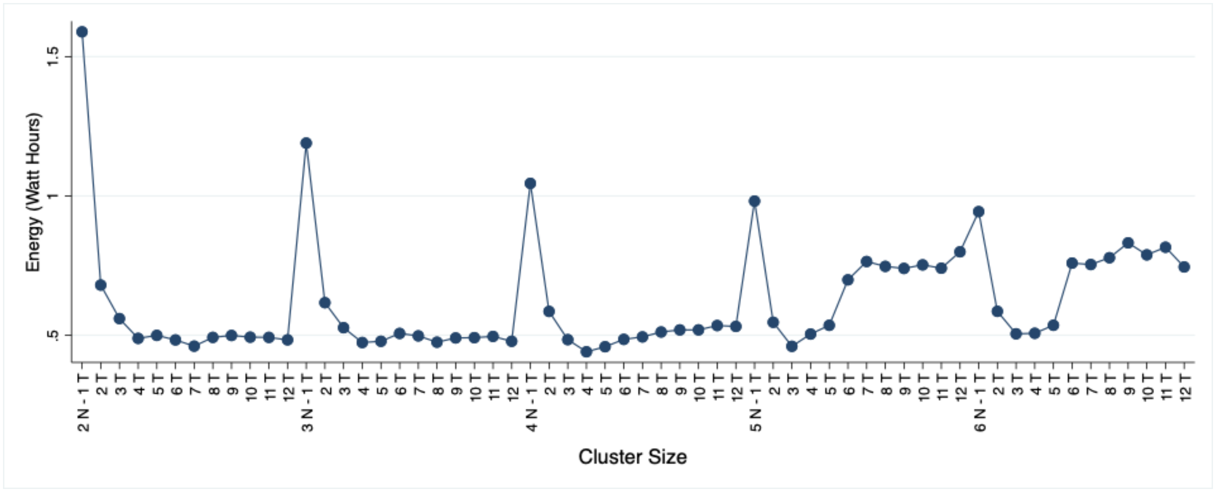

Figure 13: Energy Consumption of Matrix Multiplication (3000x3000 matrix)

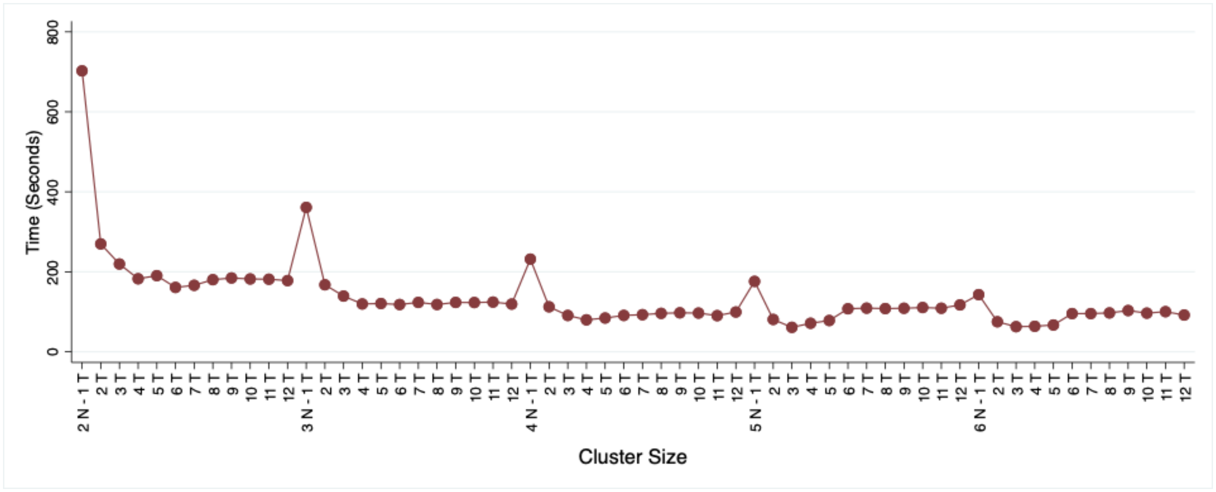

Figure 14: Time Consumption of Matrix Multiplication (3000x3000 matrix)

used to show the portability of our framework as well as its compatibility with FORTRAN libraries.

For this experiment, the framework was provided with an algorithm file which in turn was provided a set of 288, 000 data points to sort into 48 clusters. The algorithm ran iteratively until the cluster centers did not change their position.

This algorithm was chosen as researchers have tried to parallelise the computation but could not gain any significant improvements [28]. Same conclusions were obtained from an energy perspective. The algorithm does not improve or in some cases worsens the performance as well as energy consumption when adding new nodes.

Figure 17 illustrates the results of using the FEPAC framework to evaluate the k-means algorithm on increasing sizes of cluster. Figure 17 shows the energy and the run-time performance data collected by the framework executing k-means clustering algorithm. The left and right Y axis comprises of energy consumption and run-time of algorithm respectively. The close relation between the run-time and the energy consumption can be used to conclude different limitations and functionalities of the algorithm.

The results can be used to conclude a number of things. It clearly showcases the bottleneck in the k-means algorithm: communication and synchronisation. Also, it shows that adding new nodes to a cluster degrades the performance of the algorithm. This result can help researchers in choosing another algorithm or other ways to improve on algorithm for their computation needs.

5 DISCUSSION

The framework allows scientists to evaluate the performance of distinct parallelisable algorithms on an range of cluster computing configurations. The experimental evaluation has demonstrated that

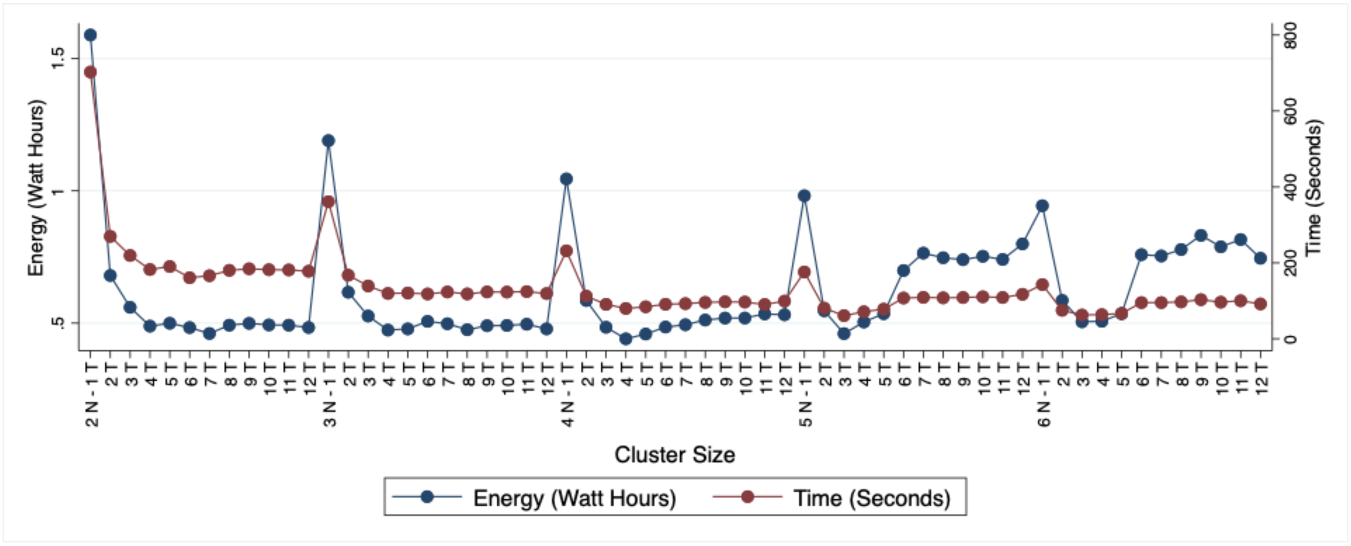

Figure 15: Matrix Multiplication and Combined Consumption (3000x3000 matrix)

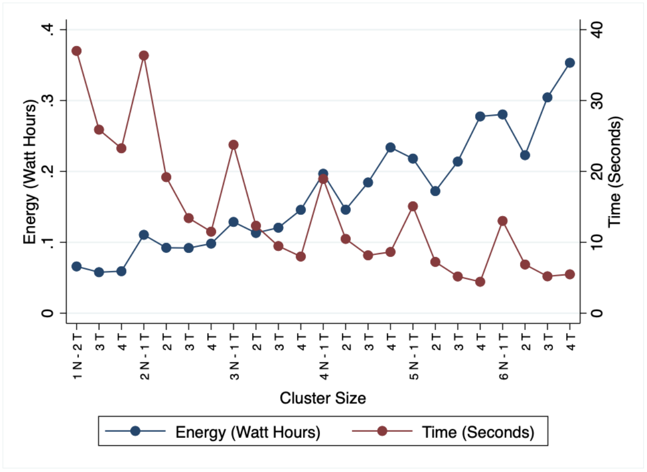

Figure 16: Energy Consumption and Execution Time for OpenCV Algorithm (5 Images, 6 Filters)

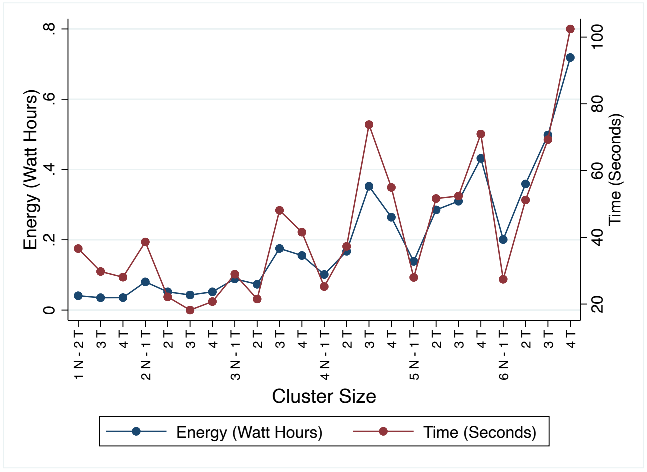

Figure 17: Energy Consumption and Execution Time for k-means Algorithm ($288,000$ points, 48 clusters)

the FEPAC framework can be used for evaluating the performance of distinct algorithms and different scales of cluster.

Three different algorithms have been evaluated using the framework to show its functionality. These algorithms have been taken from distinct research domains and comprises different programming languages to demonstrate the compatibility and reliability of the framework. Each experiment presented in Section 2.1 demonstrated the framework's ability to meet the requirements devised in Section 3 - to produce data that exposes the impact of a cluster configuration on performance or performance per watt for a particular algorithm.

The framework can identify that no significant improvement was achieved in the performance over a number of threads on each node (Figure 15). It can also be used to predict the computation time and energy consumption for algorithms as the number of cluster nodes is increased (Figures 14, 17 and 16).

In addition to this core aim, the framework also supports the identification of significant features of a computation workload. The framework enables identification of bottlenecks in a computation. The framework identifies the cache memory interference (Figure 14) in matrix multiplication and communication and synchronisation as a bottleneck in k-means (Figure 17). The framework show that OpenCV is not optimised for small data-sets (Figure 16).

6 CONCLUSIONS AND FUTURE WORK

This paper has argued that the lack of a generic programming support for the evaluation of clusters based on their energy consumption is harming the motivation for research and development in the field of Energy aware computing. It has demonstrated the design and implementation of a framework for supporting the evaluation of parallel computation based on a number of factors including its energy consumption and performance. A 6-node SBC-based cluster was used allowing configuration by the number of threads each node is using for the computation. Three different algorithms have been evaluated using the framework on different cluster configurations. The framework presented in this paper can be used to evaluate clusters with any number of nodes and configurations.

The work presented in this paper can be expanded in a number of different ways. This paper has focused on the core algorithms of parallel computation, however, the majority of scientific work utilises higher-level engines. Workflows allow scientists to build a computation graph of interdependent tasks tasks that can be used for complex computation. The work presented in this paper can be expanded to include support for workflow engines. Containers, such as Docker, allow pseudo virtualisation and application packaging for workload deployment. Containers are packages that include a run-time environment, executable code and library dependencies. The work presented in this paper can be expanded to include evaluation of container-based executions. The process of interpreting data could also be automated by using machine learning to train the framework to provide a predictive response based on user need.

REFERENCES

[1] Nikilesh Balakrishnan. 2012. *Building and benchmarking a low power ARM cluster.* Master's thesis. University of Edinburgh.

[2] Philip J. Basford, Steven J. Johnston, Colin S. Perkins, Tony Garnock-Jones, Fung Po Tso, Dimitrios Pezaros, Robert D. Mullins, Eiko Yoneki, Jeremy Singer, and Simon J. Cox. 2020. Performance Analysis of Single Board Computer Clusters. *Future Generation Computer Systems* 102 (Jan. 2020), 278–291.

[3] Ken A. Berman and Jerome Paul. 1996. *Fundamentals of Sequential and Parallel Algorithms.* PWS Publishing Co.

[4] Dhruba Borthakur. 2007. The hadoop distributed file system: Architecture and design. *Hadoop Project Website* 11, 2007 (2007), 21.

[5] Dinh-Mao Bui, YongIk Yoon, Eui-Nam Huh, SungIk Jun, and Sungyoung Lee. 2017. Energy efficiency for cloud computing system based on predictive optimization. *J. Parallel and Distrib. Comput.* 102 (2017).

[6] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* 41 (2011).

[7] Michael F Cloutier, Chad Paradis, and Vincent M Weaver. 2016. A raspberry pi cluster instrumented for fine-grained power measurement. *Electronics* 5 (2016).

[8] Javier Conejero, Omer Rana, Peter Burnap, Jeffrey Morgan, Blanca Caminero, and Carmen Carrión. 2016. Analyzing Hadoop power consumption and impact on application QoS. *Future Generation Computer Systems* 55 (2016).

[9] Eugen Feller, Lavanya Ramakrishnan, and Christine Morin. 2015. Performance and energy efficiency of big data applications in cloud environments: A Hadoop case study. *J. Parallel and Distrib. Comput.* 79 (2015).

[10] Joseph JéJé. 1992. *An Introduction to Parallel Algorithms.* Reading, MA: Addison-Wesley.

[11] Chao Jin, Bronis R de Supinski, David Abramson, Heidi Poxon, Luiz DeRose, Minh Ngoc Dinh, Mark Endrei, and Elizabeth R Jessup. 2017. A survey on software methods to improve the energy efficiency of parallel computing. *The International Journal of High Performance Computing Applications* 31 (2017).

[12] Tarandeep Kaur and Inderveer Chana. 2015. Energy efficiency techniques in cloud computing: A survey and taxonomy. *ACM computing surveys* 48 (2015).

[13] Gabor Kecskemeti, Wajdi Hajji, and Fung Po Tso. 2017. Modelling low power compute clusters for cloud simulation. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP).*

[14] KR Krish, M Safdar Iqbal, M Mustafa Rafique, and Ali R Butt. 2014. Towards energy awareness in hadoop. In *2014 Fourth International Workshop on Network-Aware Data Management.*

[15] Kevin Lee, Norman W Paton, Rizos Sakellariou, Ewa Deelman, Alvaro AA Fernandes, and Gaurang Mehta. 2009. Adaptive workflow processing and execution in pegasus. *Concurrency and Computation: Practice and Experience* 21, 16 (2009), 1965–1981.

[16] Piotr Luszczek, Jack J. Dongarra, David Koester, Rolf Rabenseifner, Bob Lucas, Jeremy Kepner, John McCalpin, David Bailey, and Daisuke Takahashi. 2005. Introduction to the HPC Challenge Benchmark Suite. (2005).

[17] Ketan Maheshwari, Eun-Sung Jung, Jiayuan Meng, Vitali Morozov, Venkatram Vishwanath, and Rajkumar Kettimuthu. 2016. Workflow performance improvement using model-based scheduling over multiple clusters and clouds. *Future Generation Computer Systems* 54 (2016), 206–218.

[18] Dusan Markovic, Dejan Vujicic, Dragana Mitrovic, and Sinisa Randic. 2018. Image Processing on Raspberry Pi Cluster. *International Journal of Electrical Engineering and Computing* 2 (2018).

[19] G. E. Moore. 2006. Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter* 11 (2006).

[20] Alberto Nunez, Jose Luis Vazquez-Poletti, Agustin C Caminero, Jesus Carretero, and Ignacio Martin Llorente. 2011. Design of a new cloud computing simulation platform. In *International Conference on Computational Science and Its Applications.*

[21] R. Panda and L. K. John. 2017. Proxy Benchmarks for Emerging Big-Data Workloads. In *2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT).*

[22] Dimitrios Papakyriakou, Dimitra Kottou, and Ioannis Kostouros. 2018. Benchmarking Raspberry Pi 2 Beowulf Cluster. *International Journal of Computer Applications* 975 (2018).

[23] Sumit Patel, MB Potdar, and Bhadreshsinh Gohil. 2015. A survey on image processing techniques with OpenMP. *International Journal of Engineering Development and Research* 3 (2015).

[24] G Pomaska. 2019. Stereo Vision Applying OpenCV and Raspberry Pi. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* XLII-2/W17 (2019).

[25] Basit Qureshi and Anis Koubaa. 2017. Power efficiency of a SBC based Hadoop cluster. In *International Conference on Smart Cities, Infrastructure, Technologies and Applications.*

[26] Romi Fadillah Rahmat, Triyan Saputra, Ainul Hizriadi, Tifani Zata Lini, and Mahyuddin KM Nasution. 2019. Performance Test of Parallel Image Processing Using Open MPI on Raspberry PI Cluster Board. In *2019 3rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM).*

[27] Max Roser and Hannah Ritchie. 2013. Technological Progress. *Our World in Data* (2013).

[28] João Saffran, Gabriel Garcia, Matheus A Souza, Pedro H Penna, Márcio Castro, Luís FW Góes, and Henrique C Freitas. 2016. A low-cost energy-efficient Raspberry Pi cluster for data mining algorithms. In *European Conference on Parallel Processing.*

[29] Nick Schot. 2015. Feasibility of raspberry pi 2 based micro data centers in big data applications. In *23th University of Twente Student Conference on IT.*

[30] Kathiravan Srinivasan, Chuan-Yu Chang, Chao-Hsi Huang, Min-Hao Chang, Anant Sharma, and Avinash Ankur. 2018. An Efficient Implementation of Mobile Raspberry Pi Hadoop Clusters for Robust and Augmented Computing Performance. *Journal of Information Processing Systems* 14 (2018).

[31] Nidhi Tiwari, Umesh Bellur, Santonu Sarkar, and Maria Indrawan. 2016. Identification of critical parameters for MapReduce energy efficiency using statistical Design of Experiments. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW).*

[32] Fung Po Tso, David R White, Simon Jouet, Jeremy Singer, and Dimitrios P Pezaros. 2013. The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures. In *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops.*

[33] A. Uta, S. Au, A. Ilyushkin, and A. Iosup. 2018. Elasticity in Graph Analytics? A Benchmarking Framework for Elastic Graph Processing. In *2018 IEEE International Conference on Cluster Computing (CLUSTER).*

[34] Yuqing Zhu, Jianfeng Zhan, Chuliang Weng, Raghunath Nambiar, Jinchao Zhang, Xingzhen Chen, and Lei Wang. 2014. BigOP: Generating Comprehensive Big Data Workloads as a Benchmarking Framework. In *Database Systems for Advanced Applications*, Sourav S. Bhowmick, Curtis E. Dyreson, Christian S. Jensen, Mong Li Lee, Agus Muliantara, and Bernhard Thalheim (Eds.). Springer International Publishing, Cham, 483–492.