electronics

MDPI

Article

Linear Weighted Regression and Energy-Aware Greedy Scheduling for Heterogeneous Big Data

Suresh Kallam [1], Rizwan Patan [2], Tathapudi V. Ramana [3] and Amir H. Gandomi [4,*]

[1] Department of Computer Science & Engineering, Sree Vidyanikethan Engineering College, Tirupati-517102, India; sureshkallam@gmail.com
[2] Department of Computer Science and Engineering, Velagapudi Ramakrishna Siddhartha Engineering College, Vijayawada 520007, India; prizwan5@gmail.com
[3] Federal Technical & Vocational Education and Training Institute (TVET), Addis Ababa 1000, Ethiopia; venkataramana.t@gmail.com
[4] Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia
* Correspondence: gandomi@uts.edu.au; Tel.: +61-(02)-9514-5081

Abstract: Data are presently being produced at an increased speed in different formats, which complicates the design, processing, and evaluation of the data. The MapReduce algorithm is a distributed file system that is used for big data parallel processing. Current implementations of MapReduce assist in data locality along with robustness. In this study, a linear weighted regression and energy-aware greedy scheduling (LWR-EGS) method were combined to handle big data. The LWR-EGS method initially selects tasks for an assignment and then selects the best available machine to identify an optimal solution. With this objective, first, the problem was modeled as an integer linear weighted regression program to choose tasks for the assignment. Then, the best available machines were selected to find the optimal solution. In this manner, the optimization of resources is said to have taken place. Then, an energy efficiency-aware greedy scheduling algorithm was presented to select a position for each task to minimize the total energy consumption of the MapReduce job for big data applications in heterogeneous environments without a significant performance loss. To evaluate the performance, the LWR-EGS method was compared with two related approaches via MapReduce. The experimental results showed that the LWR-EGS method effectively reduced the total energy consumption without producing large scheduling overheads. Moreover, the method also reduced the execution time when compared to state-of-the-art methods. The LWR-EGS method reduced the energy consumption, average processing time, and scheduling overhead by 16%, 20%, and 22%, respectively, compared to existing methods.

Keywords: MapReduce; distributed file system; big data; linear weighted regression; energy-aware greedy scheduling; heterogeneous environment

Citation: Kallam, S.; Patan, R.; Ramana, T.V.; Gandomi, A.H. Linear Weighted Regression and Energy-Aware Greedy Scheduling for Heterogeneous Big Data. *Electronics* **2021**, *10*, 554. https://doi.org/10.3390/electronics10050554

Academic Editor: Rashid Mehmood

Received: 22 December 2020
Accepted: 8 February 2021
Published: 26 February 2021

1. Introduction

One of the primary insightful ways to store and compute big data is via distributed and parallel processing. A large number of data are being assembled every day from web authoring, digital media, scientific instruments, etc. However, this is complicated by limitations, with regard to the software and in the research community, when it comes to the effectiveness with which the big data are stored, queried, analyzed and forecasted. An important computational framework for large-scale data processing is known as MapReduce. Existing applications of MapReduce aid in data locality along with robustness.

Previously, a novel Map Task Data Locality-based Scheduler (MTDLS) was presented [1] with the prime objective of allocating input data blocks to nodes based on their processing capabilities. Depending on their computing capabilities in the heterogeneous Hadoop Cluster, the map scheduling and reduction tasks were performed to the

corresponding nodes. The nodes with fast processing capabilities were comparatively allocated to more tasks than the nodes with slow processing capabilities.

This scheduler has been evaluated using different workloads from the high benchmark suite. With this process, the scheduler also reduces the job execution time with improved data locality. Though the job execution time is reduced with improved data locality, the average processing time is high. Thus, in this work, to reduce the average processing time, first, tasks were deliberately chosen for the assignment, and then, the best available machine was selected in an energy-efficient manner. In this way, the average processing time should be reduced in a large heterogeneous environment, which in turn should enhance the MapReduce performance.

Several methods have been presented to implement the Apriori algorithm on the MapReduce framework. The Apriori algorithm performs the frequent item set mining and association rule learning over databases. However, only a few methods have attained its performance enhancement. Thus, improved MapReduce-based Apriori algorithms called Variable Fixed Passes Combined-counting (VFPC) and Elapsed Time-based Dynamic Passes Combined-counting (ETDPC) were previously proposed [2]. In addition, the number of passes is reduced by skipping the pruning step in certain passes by designing the Optimized-VFPC and Optimized-ETDPC algorithms.

A quantitative analysis revealed that despite the increased number of candidates, the execution time is notably less and is also scalable with an increasing number of nodes. In addition, the energy consumed during the optimization process is less concentrated [2]. To address this issue, in this work, an energy efficiency-aware greedy scheduling algorithm is designed, which focuses not only on the energy being consumed but also on the scheduling overhead involved.

Distributed parallel methods and mechanisms are frequently used to achieve the performance and scalability requirement for clustering large datasets. In a previous study [3], a parallel k-means algorithm with the MapReduce programming model was used in document clustering to minimize the processing time [4]. Recently, the utilization of big data has increased due to the fact that it has performed very efficiently in several fields, including social media and E-commerce transactions. A comprehensive overview of big data is provided in another study [5], with a brief analysis of the comparative study of job-scheduling algorithms.

Currently, serverless computing and Functions as a Service (FaaS) have evolved as an execution pattern wherein the user does not manage the servers. Previously [6], a high-performance serverless architecture was created to execute MapReduce jobs as the storage backend, therefore reducing the execution time involved. However, the storage overhead was high. Thus, to address this issue, a dynamically adaptable block IO(Input/Output) scheduling scheme was presented [7]. This, in turn, reduced the MapReduce workloads without compromising the Service Level Agreements (SLA).

The extensive use of new materials and methods results in the generation of a large data volume. With the inception of the "4Vs", i.e., volume, velocity, variety and veracity, which specialize in the complexity involved in big data, good performance is said to be ensured based on the volume of transactions being handled. In a previous study [8], a fine-grained parallelization model for Data Warehouse (DW) was designed, which provided several means of parallelization at both the functionality and elementary levels.

However, the resources required in a heterogeneous environment are higher with the nature and type of job involved. A Dynamic Grouping Integrated Neighboring Search [9] provides a means for a balanced resource utilization and also improves the data locality involved in the heterogeneous environment. In addition, another decision tree classification algorithm was introduced [10], which is based on the programming framework of MapReduce, with the objective of improving accuracy in a big data environment.

Recently, research on big data using MapReduce in a heterogeneous environment has become increasingly popular. Certain notable issues that still need to be solved are the type of heterogeneity, enhancing the MapReduce performance, the storage distribution, the

retrieval performance and the time factor evaluation. As far as the big data retrieval system is concerned, designing effective algorithms to swiftly and precisely acquire the necessitated content with a minimum energy consumption has become a salient issue. In addition, users might desire obtaining different types of information, such as those that consider their query objectives and necessitate a cross-type of information characterization, with the minimum job execution time involved. Moreover, due to the scalable nature of data, is important to determine how to minimize the time cost evaluation in a heterogeneous environment, in addition to reducing the scheduling overhead.

2. Contribution

To solve the problems described above, we built an optimized linear weighted regression with MapReduce algorithm to reduce the average processing time for a heterogeneous environment. Then, an energy efficiency-aware greedy scheduling algorithm was designed with the objective of reducing the energy that was consumed during the selection of the best available machine.

The major contributions of this paper include the following:

- Algorithms that support big data in a heterogeneous environment are presented. Any type of big data can be retrieved based on linear weighted regression with MapReduce algorithms.
- The intermediate key-value pairs are stored via metadata together with the map tasks and the reduce tasks and do not straightforwardly process large-sized big data.
- The energy-efficiency-aware greedy scheduling algorithm, together with the MapReduce framework, is designed to parallel store data and process the retrieval for big data applications in heterogeneous environments.
- This is a proposal of the scheduling model for reducing energy consumption during the scheduling of the machine by improving the scheduling overhead and also minimizing the job execution time. Thus, the proposed approach enhances the MapReduce performance in heterogeneous environments in an energy-aware manner.

The remainder of this paper is organized as follows. Section 2 explains the work related to the heterogeneous data retrieval and the big data processing methods. Section 3 describes the system model and background description. Section 4 presents the solution to the problem, the algorithm description and the complexity analysis with a detailed mathematical model. A performance evaluation is provided in Section 5, which presents detailed experimental results. The conclusions are provided in Section 6.

3. Related Work

Over the past few years, one of the effective platforms for processing large unstructured data has been via system logs that implement Extract Transform Load models and measuring web indexes. Because big data analytics necessitate distributed computing at a higher rate, specifically involving thousands to millions of machines, one of the most extensive barriers is the practice of accessing big data processing in small and medium business establishments.

Even though MapReduce is cost-effective, with a heterogeneous nature, it has evolved to present a significant limitation when it comes to performance. To address this issue, a self-adaptive task pruning approach was previously presented [11] to reduce the average job completion time and to also minimize the Input Output (IO) rate by applying the k-means clustering algorithm.

In the method, the search intentions of the users are conveyed via semantic means, and this model has thus become a paramount tool for information retrieval. To assist the semantic-based multimedia retrieval in a big data environment, an optimized Semantic Ontology Retrieval (SOR) algorithm was previously presented [12], which utilized big data processing tools with the purpose of storing and retrieving heterogeneous multimedia data. For example, a large number of multi-dimensional data have been generated via geoscience observations. However, it is cumbersome for geoscientists to perform these

tasks, because the processing of a massive number of data requires a lot of time. In addition, due to the data-intensive nature, the data analytics require complicated mechanisms and several tools.

To address the above-mentioned issues, a scientific workflow framework was proposed [13] by leveraging cloud computing, MapReduce and Service-Oriented Architecture (SOA). With this framework, the data processing time was minimized. In addition, another MapReduce Task Scheduler using Dynamic Calibration was designed [14] in a heterogeneous MapReduce environment.

Thus, heterogeneous architectures have transpired with encouraging results to improve the energy efficiency by providing each user with the chance to run on a core that equates resource requirements more efficiently compared to the size-fits-all processing node. In a previous study [15], k-means, K Nearest Neighbor, Support Vector Machine and Naive Bayes were examined to measure the power efficiency and energy efficiency of the system. However, guaranteeing the Quality of Service (QoS) with a minimum resource cost is now one of the new issues in computation-intensive MapReduce computations that also involve dynamic changes. To address this issue, a new event-driven framework was developed that aimed to reduce the running cost involved in the MapReduce computation [16]. In addition, to cope with the increasing size of datasets and minimal analysis time, Hadoop MapReduce utilized parallel computations on large, distributed clusters that comprised several machines involved in the design. Furthermore, another study focused on measuring the data replication factor on the MapReduce job completion time with the aid of regression analysis, and the total job completion time was exponentially reduced [17].

Due to the challenges involved in big data and analytical methods, a brief critical analysis was conducted [18]. However, the analysis was not very rigorous. To address this issue, parallelizing the Back Propagation Neural Network using MapReduce was performed in another study [19], with the objective of not only improving the efficiency in terms of precision but also refining the classification results. In addition, the heterogeneous architecture in big data processing using the MapReduce paradigm was also presented [20].

In [21,22], a power efficiency model was proposed for measuring the power flow approach based on the S-iteration process and a robust power flow algorithm based on bulirsch–stoer method. As part of the proposed study, this paper helps to measure the power efficiency effectively. In [23,24], different platformers are shown to measure energy efficiency in terms of Cloud sharing environment, software-defined network and 4G/5G mobile data communication fields in terms of energy consumption and data handling, with these technologies and challenges addressed in detail. By considering these approaches evolve the proposed model in different directions by addressing all the challenges.

In summary, although the methods presented above attain a good performance, difficulties, such as the average processing time and scheduling overhead, prevent these methods from being widely applied. Our method, namely Linear Weighted Regression and Energy-aware Greedy Scheduling (LWR-EGS), provides a different approach for deliberately selecting the tasks for assignment and then selecting the best available machine (i.e., resource) for them in an energy-efficient manner. The experimental results presented here indicate the effectiveness of the proposed algorithm.

4. Methodology

This section presents the main contribution of the paper, including the design and development of the proposed method. This method bridges the gap between choosing tasks for an assignment and selecting the best available machine, improving the MapReduce performance, and providing an algorithm suite in MapReduce. In the proposed LWR-EGS method, we address the limitations of the MapReduce framework that are faced in a heterogeneous environment and propose a Linear Weighted Regression-based model for optimal resource scheduling. Furthermore, the main idea and motivation of the LWR-EGS method are to minimize the average processing time and energy consumption while selecting the best available machine by implementing them concurrently. The following

sections formalize all the concepts in detail by starting a system model, followed by a problem description and an elaborate description of the proposal.

4.1. System Model

To process large-scale datasets in a distributed manner, one of the most prevalent frameworks is MapReduce. A user's request for data processing under the MapReduce framework is referred to as a job J. The job J is expressed as two functions Map M and Reduce R. The input data for a job J are split into data chunks and are stored in a distributed manner across the cluster via a distributed file system. A user submits job J, and on the other hand, the MapReduce framework splits job J into multiple map M_i tasks and reduces R_i tasks as follows.

Each data chunk from the corresponding user input data relates to one map task that studies the data chunk and relates the map function to the data chunk. Then, intermediate data are generated. The intermediate data are represented in the form of key-value $K - V$ pairs. These key-value

$K - V$ pairs are stored on local disks in machines that process the map tasks. The set of all the probable keys K of the intermediate data is split into different subsets. Each of the different subsets coincides with a reduce task. The reduce task R retrieves the intermediate key-value pairs from all the map tasks and applies to the reduce function. By doing this, the reduce task R integrates the intermediate data and generates the results. The above-mentioned representations are provided below:

$$\begin{aligned} T = \; & MAPREDUCE\,(Input)\langle k_1,\, v_1\rangle \\ & \rightarrow\; MAP \rightarrow\; \langle k_2,\, v_2\rangle \\ & \rightarrow\; REDUCE \rightarrow\; \langle k_3,\, v_3\rangle \end{aligned} \tag{1}$$

From Equation (1), the MapReduce of three different key-value pairs is performed. Here, T indicates a task, k_1 indicates a key and v_1 indicates a value.

4.2. Problem Description

MapReduce is an abstraction to arrange immense parallel tasks with Hadoop, and it is an open-source implementation that is a software environment specifically meant for the distributed processing of enormous data sets over several machines. The main concept behind the design of MapReduce is to map the given input data set into a group of key and value pairs and then minimize all the pairs with the same key. The MapReduce is comprised of a set of maps and reduce tasks. The MapReduce job is split into two processes, the Map process, and the Reduce process, which perform the map tasks and reduce tasks, respectively. The Reduce process is said to be initiated once the execution of the Map process is accomplished. First, the Map process (via script) obtains certain input data and maps them to the $\langle Key,\, Value\rangle$ pairs. Then, it processes the intermediate data. Finally, the reduce process (via script) obtains these intermediate data as $\langle Key,\, Value\rangle$ pairs and reduces the pairs with the same key into a single pair.

The problem is to first deliberately select the tasks for the assignment and then to select the best available machine (i.e., resource) for them. In this manner, the optimization of the resources takes place. Second, with the optimized resources assigned to each of them (i.e., resources), the overall energy consumption of all the tasks is reduced within the stipulated time in heterogeneous environments [25]. Here, the resource optimization is based on prior knowledge about the projected energy consumption and the projected processing time of each task while executing each machine. In addition, assigning a task to a position here refers to positioning the data on the machine to which the position belongs and then running the corresponding script (i.e., either the map process or the reduce process).

4.3. Integer Linear Weighted Regression (ILWR) model

The Integer Linear Weighted Regression model was used to predict the time span of the map, schedule and reduce phases. The objective was to design an optimized resource

scheduler that was responsible for deciding which task was to be executed at which time and on which machine by considering both the central processing unit (CPU) and the memory. The most common objective of the optimized resource scheduling was to reduce the job completion time optimally by properly allocating the tasks to the processors using [26]. With this Integer Linear Weighted Regression (ILWR) model, the tasks were selected deliberately for assignment, and with this, the best available machine (i.e., resource) was selected for them. In this manner, resource optimization occurred. Figure 1 shows the block diagram of the ILWR model.

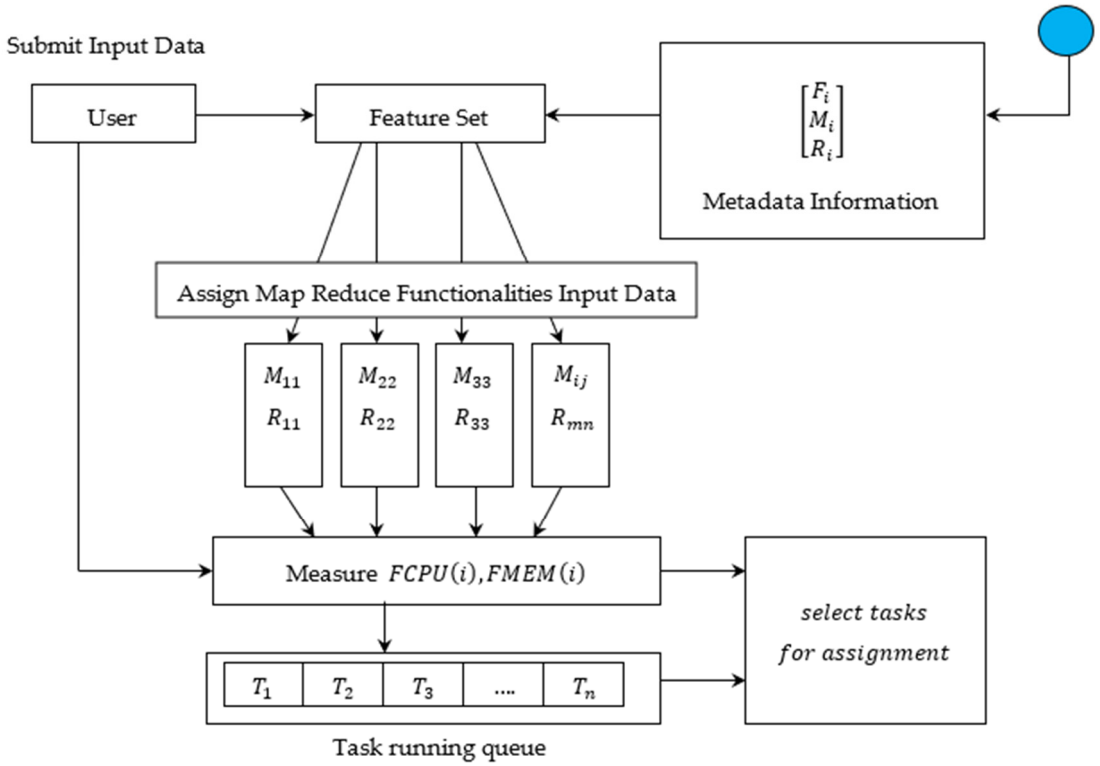

Figure 1. Block diagram of the Integer Linear Weighted Regression (ILWR) model.

The feature sets utilized in the ILWR model are listed below. First, the tasks are selected for assignment using an ILWR model. Then, the file size F_i, the number of reduces R_i and the number of machines involved in the processing M_i to denote the *ith* sample p with n number of samples are input. The above-mentioned feature sets are expressed below.

$$p_i = \begin{bmatrix} F_i \\ M_i \\ R_i \end{bmatrix}, \; i = 1, 2, \ldots, n \tag{2}$$

From (2), F_i indicates a file size, M_i number of machines and R_i number of reduces. Next, a matrix P was defined to represent all the training data, and a vector Q was used to represent the time, which correlated to the sample p_i. Then, the matrix P and the vector Q were represented as given below.

$$P = [p_1, \; p_2, \; \ldots, p_i] \tag{3}$$

$$Q = [q_1, \; q_2, \; \ldots, q_i]^{Pred} \tag{4}$$

For the prediction of $Pred_{phase}$, the weight for each sample was measured to identify the most similar tasks. Then, the distance measure was used in the following expression:

$$D\left(p_{sample}, \; p_{pred}\right) = \sqrt{\left(F_{sample} - F_{pred}\right)^2 + \left(M_{sample} - M_{pred}\right)^2 + \left(R_{sample} - R_{pred}\right)^2} \tag{5}$$

In the above Equation (5), p_{sample} corresponds to the sample point and p_{pred} corresponds to the prediction point. Their feature values were correspondingly used to measure the distance between them. With the measured distance, selecting the best available machine with the objective of optimizing the resource was performed via the integer linear model used based on [27]. The integer linear model was formulated as given below, where i represents the ID number of a MAP task and j represents the ID number of a $REDUCE$ task. In addition, m represents the ID number of a MAP position, and n represents the ID number of a $REDUCE$ position. Moreover, T_l represents the time limit for completing all the tasks for an assignment, with T^{MAP} and T^{REDUCE} representing the deadlines for the completion of all of the MAP tasks and $REDUCE$ tasks, respectively.

$$M_{ij} = \begin{cases} 1, \; if \; MAP \; task \; i \; is \; assigned \; to \; MAP \; position \; j \\ -1, \; Otherwise \end{cases} \tag{6}$$

$$R_{mn} = \begin{cases} 1, \; if \; MAP \; task \; m \; is \; assigned \; to \; MAP \; position \; n \\ -1, \; Otherwise \end{cases} \tag{7}$$

From Equations (6) and (7) above, two sets of variables were defined for determining which position should run which tasks, and these were denoted by M_{ij} and R_{mn}, respectively, with the constraints as given below.

$$M_{ij} = 1, \; \forall \; i \; \in I \tag{8}$$

$$R_{mn} = 1, \; \forall \; m \; \in M \tag{9}$$

The Equation (8) above ensures that each MAP task is allocated to one and only one MAP position, and all the tasks are hence said to be allocated. Similarly, Equation (9) ensures that each $REDUCE$ task is allocated to one and only one $REDUCE$ position, and all the tasks are hence said to be reduced. After that, the free memory and CPU usage for every task are measured using the expressions given below.

$$FCPU(i) = \frac{100 - UCPU(i) * CPU(i)}{100} \tag{10}$$

$$FMEM(i) = \frac{(100 - UMEM(i) * MEM(i))}{100} \tag{11}$$

From Equations (10) and (11), $CPU(i)$ refers to the processing capacity of the ith task, $MEM(i)$ refers to the RAM capacity of the ith task, $UCPU(i)$ refers to the percentage of CPU used for the ith task and $UMEM(i)$ refers to the percentage of memory used for the ith task. Finally, the prediction phase, $Pred_{phase}$, of the new instance for the optimal allocation of resources process is used [28] and is expressed as follows:

$$Pred_{phase} = M_{ij}R_{mn}[FCPU(i) \; \cup \; FMEM(i)] + \; D\Big(p_{sample}, \; p_{pred}\Big) \tag{12}$$

The proposed model presents an optimal resource scheduler that dynamically distributes the position and map tasks to a machine that possesses the maximum processing capacity in the heterogeneous environment, as shown in Algorithm 1.

Algorithm 1 Proposed algorithm for optimal resource scheduling

Input: Set of input tasks to be assigned
Output: Optimized resource scheduling
1: **For** each file size F_i, reduces R_i and machines M_i do
2: Calculate the distance $D\left(p_{sample},\ p_{pred}\right)$ between the sample point and prediction point for input file size F_i
3: Calculate the deadlines for completion of the MAP task M_{ij} and the $REDUCE$ task R_{mn}, with the constraints from (8) and (9)
4: Calculate the free CPU $FCPU(i)$
5: Calculate the free MEM $FMEM(i)$
6: Measure the prediction phase $Pred_{phase}$
7: **If** $FCPU(i)$ and $FMEM(i)$ are available then
8: assign the MAP tasks to $Pred_{phase}$
9: **Else**
10: $FCPU(i)$ and $FMEM(i)$ are not available, then
11: **Go to** step (2)
12: **End if**
13: **End for**
14: **End**

The above linear weighted regression with MapReduce algorithm has three main phases. The first phase is to estimate the training dataset and represent it in the form of a matrix and vector. The second phase is to estimate the map and reduce tasks and to set up queues accordingly. The final phase is to select the tasks for assignment via regression and resource availability while satisfying the available CPU and MEM. In this manner, the tasks for the assignments are selected optimally.

4.4. Energy-Efficiency-Aware Greedy Scheduling Algorithm

Considering the increasing significance of Hadoop MapReduce, one of the main issues faced in a heterogeneous environment while selecting the best available machine is saving energy. In this work, we present an energy-efficiency-aware greedy scheduling algorithm to choose a position for each task to minimize the total energy consumption of the MapReduce task for big data applications in heterogeneous environments without causing a significant performance loss. Figure 2 shows the block diagram of the energy-efficiency-aware greedy scheduling model.

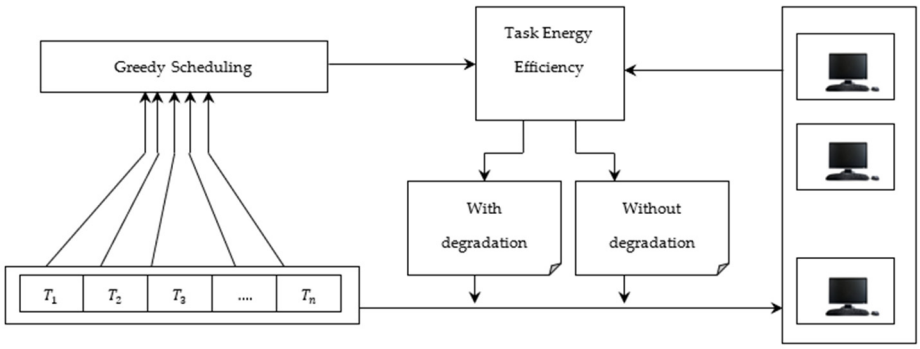

Figure 2. Block diagram of energy-efficiency-aware greedy scheduling.

The energy efficiency-aware greedy scheduling is illustrated in Figure 2 to select the best available machine. According to the definition of energy efficiency, the energy efficiency of a machine is defined by three aspects as follows:

$$EE_{Proc} = \frac{Proc\_Perf}{EC\_Proc} \tag{13}$$

$$EE_{MDIO} = \frac{MDIO}{EC\_MDIO} \tag{14}$$

$$EE_{MTR} = \frac{MTR}{EC\_MTR} \tag{15}$$

The three aspects given in Equations (13)–(15) represent the energy efficiency of a processor EE_{Proc}, the energy efficiency of the maximum disk IO rate EE_{MDIO} and the energy efficiency of the maximum transmission rate EE_{MTR}, respectively. These three aspects are obtained based on the processor performance $Proc\_Perf$, the disk IO rate $MDIO$, the transmission rate MTR to the ratio of the energy consumption of the processor EC_Proc, the energy consumption of the disk IO EC_MDIO and the energy consumption of the transmission rate EC_MTR, respectively, of the corresponding machine.

There are dissimilarities between the energy efficiency of a machine and its host server machine due to virtualization. Therefore, in this work, the machine degradation, with respect to the energy efficiency of the processing, the disk IO rate and the data transmission rate, is represented as D_{Proc}, D_{MDIO} and D_{MTR}, respectively. Thus, the energy efficiency of a machine with degradation arising due to virtualization is expressed as follows:

$$EE_{Proc}{}' = EE_{Proc} * (1 - D_{Proc}) \tag{16}$$

$$EE_{MDIO}{}' = EE_{MDIO} * (1 - D_{MDIO}) \tag{17}$$

$$EE_{MTR}{}' = EE_{MTR} * (1 - D_{MTR}) \tag{18}$$

Then, considering the task resource requirements, the machine energy features and the workload on the host machine, the energy consumption of the task run on a machine is expressed as follows:

$$Task_{Energy} = E_{Proc} + E_{MDIO} + E_{MTR} \tag{19}$$

$$APT = \sum_{i=1}^{n} T_i * Time\ (Task\_Scheduled) \tag{20}$$

$$Task_{Energy} = EE\_Proc' * T\_Proc + EE_{MDIO}{}' * T\_MIDO + EE_{Proc}{}' * T\_MTR \tag{21}$$

where

$$T\_Proc = \frac{N\_Proc}{Proc\_Perf'} \tag{22}$$

$$T\_MIDO = \frac{N\_MDIO}{MDIO'} \tag{23}$$

$$T\_MTR = \frac{N\_MTR}{MTR'} \tag{24}$$

From Equations (22)–(24), T_Proc represents the number of instructions to be processed, T_MIDO corresponds to the total number of disk data and T_MTR refers to the data transfer rate, respectively. From the above three equations, finally, considering the task resource requirements, the overall task energy is expressed as follows:

$$Task_{energy} = \left(\frac{N\_Proc}{EE\_Proc'} + \frac{N\_MDIO}{EE\_MDIO'} + \frac{N\_MTR}{EE\_MTR'} \right) \tag{25}$$

where N_Proc', N_MDIO' and N_MTR' represent the energy efficiency of a machine regarding several instructions to be processed, the IO rate and the data transfer rate, respectively. From Equation (25), it is inferred that allocating tasks to machines with higher energy efficiency is of great significance, which in turn minimizes the energy consumption. In addition, the workload is of high significance, since a high load results in greater performance degradation, which increases the energy consumption to complete a task.

The Algorithm 2 takes the machine energy efficiency and task demands into account and hence ensures an energy-efficient task scheduling. The energy-efficient task scheduling

is achieved in three phases. In the first phase, the energy efficiency of the processor without degradation is considered. Because of the heterogeneous environment, not all the machines possess an equal capacity, and every machine possesses its own processing speed, IO rate or data transfer rate. Hence, by taking these factors into consideration in the second phase, the processor energy efficiency with degradation is also considered. Finally, the energy required for task scheduling is measured. In this way, the tasks are allocated to the best available machine by considering the energy efficiency of the machine.

Algorithm 2 Proposed algorithm for energy-efficient machine selecting

Input: Set of assigned tasks
Output: Energy-efficient machine selection
1: **For** each file size F_i, reduces R_i and machines M_i do
2: Obtain the processor energy efficiency EE_{Proc}, the disk IO rate energy efficiency EE_{MDIO} and the transmission energy efficiency EE_{MTR} using (13), (14) and (15), respectively
3: Obtain the processor energy efficiency EE_{Proc}, the disk IO rate energy efficiency EE_{MDIO} and the transmission energy efficiency EE_{MTR} with degradation using (16), (17) and (18), respectively
4: Measure the energy consumption of the task run on a machine using (19) and (21)
5: Measure the overall task energy using (25)
6: **End for**

5. Experiments and Results and Performance Evaluation Model

In this paper, a low energy consumption resource scheduling method was proposed for a heterogeneous environment. Therefore, the experiments were specifically designed to measure the energy consumption along with the storage overhead and the average running time of the resource scheduling of various tasks. For the experiment platform, the Hadoop Distributed File System (HDFS) was used to construct the required storage. Establishing the experimental dataset was an important step. To measure the performance, the experiments were performed using a Kaggle dataset [29] on big city health data.

This dataset provides the status pertaining to the health of 26 of the nation's largest and most urban cities as extracted by 34 health (and six demographics-related) indicators. These measures denote some of the paramount causes of morbidity and mortality in the United States and the leading priorities of national, state and local health agencies.

The public health data were obtained from nine different overarching categories, namely HIV/AIDS, cancer, nutrition/physical activity/obesity, food safety, infectious disease, maternal and child health, tobacco, injury/violence and behavioral health/substance abuse. The columns included are the indicator category, year, indicator, gender, race, value, place, requested methodology, source, methods and notes. To ensure a fair comparison, the proposed methods were compared with two recent methods, namely MTDLS and VFPC-ETDPC.

For the comparison, the performance evaluation model was based on the following three criteria, the energy consumption, the average processing time and storage overhead.

5.1. Energy Consumption Performance Evaluation

Energy consumption was used to evaluate the effectiveness of the LWR-EGS. In the first experiment, we assessed the energy being consumed whenever the tasks were submitted by a user in a heterogeneous environment. The LWR-EGS method estimates the energy being consumed by considering certain factors, including the number of instructions being processed, the IO rate and the data transfer rate. In addition to attaining the goal of saving energy, it was of great necessity to consider a machine's energy efficiency. Assigning tasks to a high-performance machine may enhance the overall resource scheduling optimization, but at the same time, doing this results in extra energy consumption. Hence, in this work, the energy efficiency of both the machine and the machine degradation, with respect to the

energy efficiency of the processing, the disk IO rate and data transmission rate, were also considered.

$$EC = \sum_{i=1}^{n} T_i * \ Task_{energy} \tag{26}$$

The energy consumption was measured using Equation (26) above, which was based on the tasks submitted by the user for scheduling and the overall task energy. From (25), EC denotes energy consumption. $Task_{energy}$ denotes the energy consumed for each task. T_i denotes the total number of tasks. The energy consumption was measured in terms of joules J. A lower energy consumption ensured the efficiency of the method. On the other hand, a higher energy consumption showed that the resource was not being scheduled in an optimized manner. Figure 3 illustrates a graphical representation of the energy consumption using the LWR-EGS, MTDLS and VFPC-ETDPC methods.

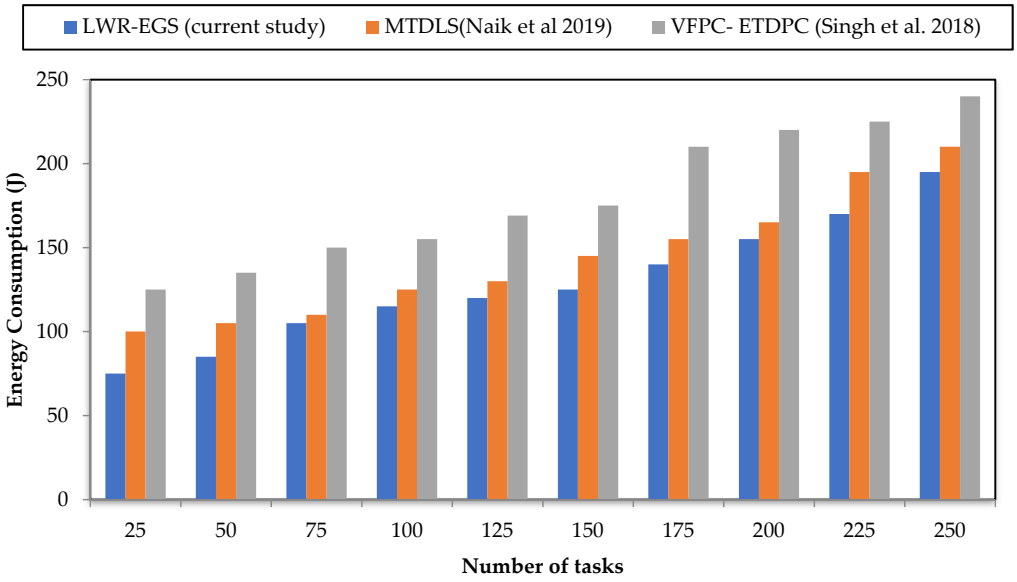

Figure 3. Energy consumption comparison.

In the simulated experiments, with the same number of tasks and machines, the initial number of tasks in the training set and those in the test set were both set to twenty-five. The x-axis in Figure 3 represents the number of tasks in the range of 25 to 250. Here, the task refers to the different indicator category obtained from a Kaggle dataset. On the other hand, the y-axis represents the different energy consumption levels obtained at the different time intervals for the different sets from the indicator categories. When considering the 75 number of tasks, the energy consumption of the LWR-EGS method, MTDLS method and VFPC-ETDPC method is 105J, 110J and 150J respectively.

A rise in energy consumption was observed with the varying numbers of tasks. By increasing the number of indicator categories or tasks, the energy being consumed for resource scheduling also increased. This was because the increase in the task being submitted by the user increased the number of resources assigned to the corresponding subsequent task, and therefore, the energy consumption also increased. However, Figure 3 shows a comparatively lower consumption level using the proposed LWR-EGS method. This occurred because of the application of the energy-efficiency-aware greedy scheduling model. Here, both the energy efficiency and greedy scheduling model were considered as two important factors for resource scheduling. Hence, the energy consumption using the LWR-EGS method was reduced by 11% compared to the MTDLS method and 20% compared to the VFPC-ETDPC method.

5.2. Performance Evaluation of the Average Processing Time

The next experiment was performed to demonstrate the effectiveness of the processed user request (i.e., resource scheduling). After asking the user to provide the request, the average processing time was measured.

$$APT = \sum_{i=1}^{n} T_i * Time\ (Task\_Scheduled) \tag{27}$$

From Equation (27), the average processing time APT was recorded based on the number of tasks provided by a user for the resource being scheduled T_i and the time consumed for scheduling $Time\ (Task\_Scheduled)$ the corresponding tasks. A lower average processing time ensured the efficiency of the method. On the other hand, a higher average processing time showed that the time consumed for scheduling the resources, on average, was higher. Figure 4 illustrates a graphical representation of the average processing time using the LWR-EGS, MTDLS and VFPC-ETDPC methods.

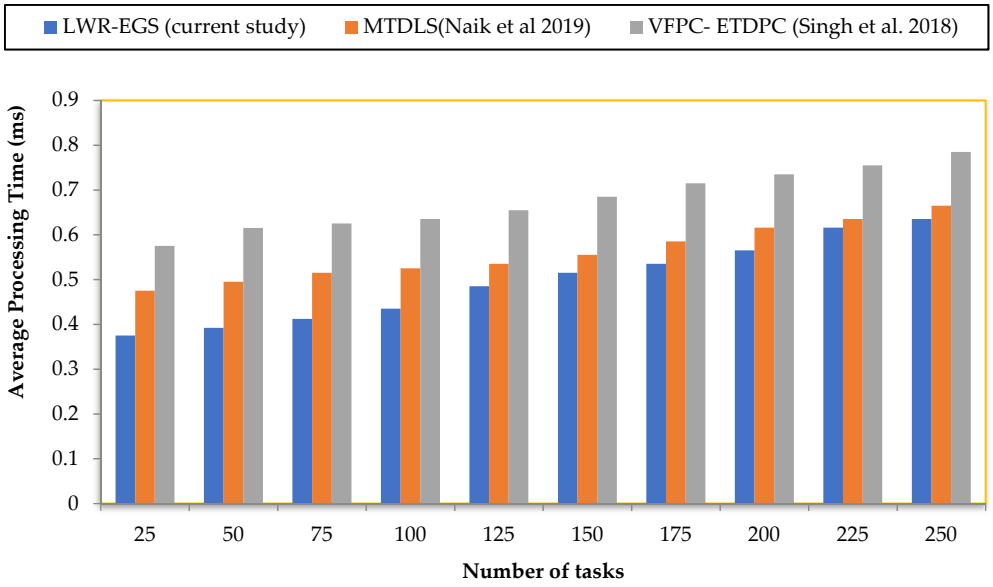

Figure 4. Comparison of the average processing time.

In Figure 4, the x-axis corresponds to the number of tasks. Here, a task refers to the indicator category. Different indicator category ranges were obtained at different time intervals. The average processing time here refers to the time taken to process the scheduling of the different tasks or jobs. From the figure, it was inferred that number of tasks was directly proportional to the average processing time. In other words, the higher the number of tasks for the assignment being provided by the user requests, the higher the time consumed in processing the subsequent user requests. However, a direct proportionality was found using the three methods. The average processing time was lower compared to MTDLS and VFPC-ETDPC, and this was evident from a simple representation.

With the number of tasks set to 25 for the experiment, the average processing time for the resource being scheduled for a single task was 0.015 ms using LWR-EGS, 0.019 ms using MTDLS and 0.023 ms using VFPC-ETDPC. In addition, the overall average processing time using MTDLS and VFPC-ETDPC were observed to be 0.375 ms, 0.475 ms and 0.575 ms, respectively. Thus, it was inferred that the average processing time using LWR-EGS was lower than MTDLS and VFPC-ETDPC. This was because of the application of the integer linear model. By applying the integer linear model, the distance between the sample point and the predicted points was used to obtain the distance, such that more similar tasks were first identified considering the CPU and memory constraints. Only after identifying the tasks were the best available machines identified for scheduling. Hence, the average

processing time was 12% lower using the LWR-EGS method compared to MTDLS and was 27% lower when compared to VFPC-ETDPC.

5.3. Performance Evaluation of the Scheduling Overhead

Finally, the third set of experiments conducted aimed to measure the effectiveness and efficiency of the method for the scheduling overhead. The scheduling overhead corresponds to the overhead incurred while scheduling the resources for the corresponding user requests that are made.

$$SO = \sum_{i=1}^{n} T_i * MEM\,[Task\_Scheduled] \tag{28}$$

From Equation (28), the scheduling overhead SO was measured based on the number of tasks requested by the user for the resource to be scheduled T_i and the memory consumed for the corresponding task to be scheduled $MEM\,[Task\_Scheduled]$. The results of the scheduling overhead are described in Figure 5. The total scheduling overhead for each number of tasks was measured. The data size, along with the number of machines to be allocated, was maintained throughout the experiment, and the tasks were added and removed after each test. The effect of the task scheduling was also measured by changing the task volume. With a constant task size at a certain threshold, the performance of the task scheduling improved by increasing the number of tasks as shown in Figure 5. In this case, when the number of tasks was 25, the overall scheduling overhead using LWR-EGS was 125 KB. In contrast, by applying MTDLS, the scheduling overhead was 175 KB, and by applying VFPC-ETDPC, it was 225 KB. Figure 5 illustrates a graphical representation of the scheduling overhead using the LWR-EGS, MTDLS and VFPC-ETDPC methods. When considering the 225 number of tasks, the scheduling overhead using MTDLS and VFPC-ETDPC methods were 390 KB, 410 KB and 365 KB, respectively.

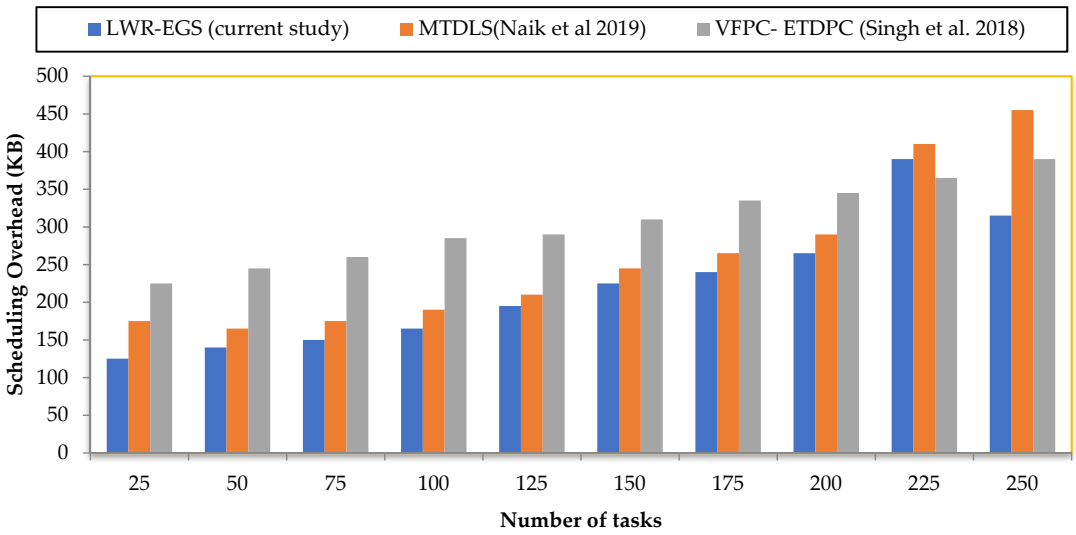

Figure 5. Comparison of the scheduling overhead.

Based on the data, it was inferred that the scheduling overhead was comparatively lower using the LWR-EGS method compared to using MTDLS and VFPC-ETDPC. The reason for the enhancement was the application of the integrated linear weighted regression with MapReduce algorithm and the energy efficiency-aware greedy scheduling algorithm. By applying these two algorithms, not only was there a deliberate selection of the tasks for the assignment, but also the best available machine was then assigned in a heterogeneous environment. In addition, when selecting the tasks for the assignment, followed by the selection of the best available machine, two factors, namely, the CPU and MEM required

and available, were also taken into consideration for the optimal resource scheduling. The comparative result analysis of the scheduling overhead is reduced by 14% and 30% using LWR-EGS method compared to existing MTDLS and VFPC-ETDPC methods.3

6. Conclusions and Future Work

Here, a low energy consumption resource scheduling method is proposed. This method is based on careful analyses of the structure of a MapReduce framework, the operating mechanism and the energy consumption problems while scheduling tasks in a heterogeneous environment. By designing an integer linear regression model, with the addition of MapReduce functionalities, the tasks to be assigned are first selected in a deliberate manner and the resources are optimally scheduled. Therefore, the indicator category from the Kaggle dataset is split into various tasks associated with different time periods, and they are processed separately. Next, by selecting the tasks to be assigned, the scheduling of the machines or resources is done by applying a greedy scheduling algorithm. In this way, by designing a resource schedule for all the tasks, the operation state of the tasks is that storage nodes are efficiently controlled to achieve energy saving. The development of creating the resource scheduling for heterogeneous big data via the MapReduce framework and an operation strategy for low energy consumption scheduling method are described in detail in this paper. The performance of the method was estimated from the energy consumption, the scheduling overhead, and the average processing time. The experimental results illustrate that compared with the state-of-the-art methods, the average processing time was drastically decreased by 20%. In addition, the results also proved that the scheduling overhead and energy consumption decreased with the LWR-EGS method compared to the state-of-the-art methods. Some limitations in the proposed LWR-EGS framework are also worth addressing in the future. (1) The integrated energy-efficient framework will be further researched considering energy-efficient scheduling, data distribution and replication dependencies factors. (2) The parts of scheduler and node management are considered in this paper, but the data distribution and replication dependencies are not considered.

Author Contributions: Conceptualization, S.K. and R.P.; data curation. S.K.; formal analysis, R.P.; methodology, S.K. and R.P.; project administration, T.V.R.; resources, R.P.; software, S.K.; supervision, R.P. and A.H.G.; validation, T.V.R. and A.H.G.; writing—original draft preparation, S.K.; writing—review and editing, R.P. and A.H.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in Kaggle at, reference number [21].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Naik, N.S.; Negi, A.; B.R., T.B.; Anitha, R. A data locality based scheduler to enhance MapReduce performance in heterogeneous environments. *Futur. Gener. Comput. Syst.* **2019**, *90*, 423–434. [CrossRef]
2. Singh, S.; Garg, R.; Mishra, P.K. Performance optimization of MapReduce-base d Apriori algorithm on Hadoop cluster. *Comput. Electr. Eng.* **2018**, *67*, 348–364. [CrossRef]
3. Wang, W.; Zhu, K.; Ying, L.; Tan, J.; Zhang, L. Map Task Scheduling in Map Reduce with Data Locality: Throughput and Heavy-Traffic Optimality. *IEEE/ACM Trans. Netw.* **2016**, *24*, 190–203. [CrossRef]
4. Sardar, T.H.; Ansari, Z. An analysis of MapReduce efficiency in document clustering using parallel K-means algorithm. *Future Comput. Inform. J.* **2018**, *3*, 200–209. [CrossRef]
5. Usama, M.; Liu, M.; Chen, M. Job schedulers for Big data processing in Hadoop environment: Testing real-life schedulers using benchmark programs. *Digit. Commun. Netw.* **2017**, *3*, 260–273. [CrossRef]
6. Giménez-Alventosa, V.; Moltó, G.; Caballer, M. A framework and a performance assessment for serverless MapReduce on AWS Lambda. *Future Gener. Comput. Syst.* **2019**, *97*, 227–259. [CrossRef]
7. Mishra, P.; Somani, A.K. Host managed contention avoidance storage solutions for Big Data. *J. Big Data* **2017**, 1–42. [CrossRef]

8. Bala, M.; Boussaid, O.; Alimazighi, Z. A Fine-Grained Distribution Approach for ETL Processes in Big Data Environments. *Data Knowl. Eng.* **2017**, *111*, 114–136. [CrossRef]
9. Chen, C.T.; Hung, L.J.; Hsieh, S.Y.; Buyya, R.; Zomaya, A.Y. Heterogeneous Job Allocation Scheduler for Hadoop MapReduce Using Dynamic Grouping Integrated Neighboring Search. *IEEE Trans. Cloud Comput.* **2017**, *8*, 1. [CrossRef]
10. Chen, T.; Liu, S.; Gong, D.; Gao, H. Data classification algorithm for data-intensive computing environments. *Eurasip J. Wirel. Commun. Netw.* **2017**, *2017*, 1–10. [CrossRef]
11. Cheng, D.; Rao, J.; Guo, Y.; Jiang, C.; Zhou, X. Improving Performance of Heterogeneous MapReduce Clusters with Adaptive Task Tuning. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *28*, 774–786. [CrossRef]
12. Guo, K.; Liang, Z.; Tang, Y.; Chi, T. SOR: An optimized semantic ontology retrieval algorithm for heterogeneous multimedia big data. *J. Comput. Sci.* **2018**, *28*, 455–465. [CrossRef]
13. Li, Z.; Yang, C.; Jin, B.; Yu, M.; Liu, K.; Sun, M.; Zhan, M. Enabling Big Geoscience Data Analytics with a Cloud-Based, MapReduce-Enabled and Service-Oriented Workflow Framework. *PLoS ONE* **2015**, *10*, e0116781. [CrossRef]
14. Lu, X.; Phang, K. An Enhanced Hadoop Heartbeat Mechanism for MapReduce Task Scheduler Using Dynamic Calibration. *Netw. Secur. China Commun.* **2018**, *15*, 93–110. [CrossRef]
15. Neshatpour, K.; Malik, M.; Sasan, A.; Rafatirad, S.; Mohsenin, T.; Ghasemzadeh, H.; Homayoun, H. Energy-efficient acceleration of MapReduce applications using FPGAs. *J. Parallel Distrib. Comput.* **2018**, *119*, 1–17. [CrossRef]
16. Xu, X.; Tang, M.; Tian, Y.C. QoS-guaranteed resource provisioning for cloud-based Map Reduce in dynamical environments. *Future Gener. Comput. Syst.* **2018**, *78*, 18–30. [CrossRef]
17. Shabbir, A.; Bakar, K.A.; Mohd, R.Z.R. Replication Effect over Hadoop MapReduce Performance using Regression Analysis. *Int. J. Comput. Appl.* **2018**, *181*, 1–6. [CrossRef]
18. Sivarajah, U.; Kamal, M.M.; Irani, Z.; Weerakkody, V. Critical analysis of Big Data challenges and analytical methods. *J. Bus. Res.* **2017**, *70*, 263–286. [CrossRef]
19. Liu, Y.; Jing, W.; Xu, L. Parallelizing Backpropagation Neural Network Using MapReduce and Cascading Model. *Comput. Intell. Neurosci.* **2016**, *2016*, 1–11. [CrossRef]
20. Goudarzi, M. Heterogeneous Architectures for Big Data Batch Processing in MapReduce Paradigm. *IEEE Trans. Big Data* **2019**, *5*, 1–18. [CrossRef]
21. Tostado-Véliz, M.; Kamel, S.; Jurado, F. A Robust Power Flow Algorithm Based on Bulirsch–Stoer Method. *IEEE Trans. Power Syst.* **2019**, *34*, 3081–3089. [CrossRef]
22. Tostado-Véliz, M.; Kamel, S.; Jurado, F. Power Flow Approach Based on the S-Iteration Process. *IEEE Trans. Power Syst.* **2020**, *35*, 4148–4158. [CrossRef]
23. Baker, T.; Aldawsari, B.; Asim, M.; Tawfik, H.; Maamar, Z.; Buyya, R. Cloud-SEnergy: A bin-packing based multi-cloud service broker for energy efficient composition and execution of data-intensive applications. *Sustain. Comput. Inform. Syst.* **2018**, *19*, 242–252. [CrossRef]
24. Khaleel, A.; Al-Raweshidy, H.S. Effective Routing Algorithm Based on Software Defined Networking for Big Data Applications in Data Centre Network. In Proceedings of the 2018 IEEE 16th Intl. Conf. on Dependable, Autonomic and Secure Computing, 16th Intl. Conf. on Pervasive Intelligence and Computing, 4th Intl. Conf. on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Athens, Greece, 12–15 August 2018; pp. 842–848. [CrossRef]
25. Dighriri, M.; Lee, G.M.; Baker, T. Measurement and Classification of Smart Systems Data Traffic Over 5G Mobile Networks. In *Technology for Smart Futures*; Springer Science and Business Media LLC: Berlin, Germany, 2018; pp. 195–217.
26. Lin, W.; Peng, G.; Bian, X.; Xu, S.; Chang, V.; Li, Y. Scheduling Algorithms for Heterogeneous Cloud Environment: Main Resource Load Balancing Algorithm and Time Balancing Algorithm. *J. Grid Comput.* **2019**, *17*, 699–726. [CrossRef]
27. Liu, B.; Li, P.; Lin, W.; Shu, N.; Li, Y.; Chang, V. A new container scheduling algorithm based on multi-objective optimization. *Soft Comput.* **2018**, *22*, 7741–7752. [CrossRef]
28. Zhang, F.; Ge, J.; Li, Z.; Li, C.; Wong, C.; Kong, L.; Luo, B.; Chang, V. A load-aware resource allocation and task scheduling for the emerging cloudlet system. *Futur. Gener. Comput. Syst.* **2018**, *87*, 438–456. [CrossRef]
29. Datasets. Available online: https://www.kaggle.com/datasets (accessed on 10 March 2020).