applied sciences

MDPI

Article

An Auto-Scaling Framework for Analyzing Big Data in the Cloud Environment

Rachana Jannapureddy, Quoc-Tuan Vien *, Purav Shah and Ramona Trestian

Faculty of Science and Technology, Middlesex University, The Burroughs, London NW4 4BT, UK; rachanareddy70@gmail.com (R.J.); P.Shah@mdx.ac.uk (P.S.); R.Trestian@mdx.ac.uk (R.T.)

* Correspondence: Q.Vien@mdx.ac.uk; Tel.: +44-208-411-4016

Received: 28 March 2019; Accepted: 29 March 2019; Published: 4 April 2019

Abstract: Processing big data on traditional computing infrastructure is a challenge as the volume of data is large and thus high computational complexity. Recently, Apache Hadoop has emerged as a distributed computing infrastructure to deal with big data. Adopting Hadoop to dynamically adjust its computing resources based on real-time workload is itself a demanding task, thus conventionally a pre-configuration with adequate resources to compute the peak data load is set up. However, this may cause a considerable wastage of computing resources when the usage levels are much lower than the preset load. In consideration of this, this paper investigates an auto-scaling framework on cloud environment aiming to minimise the cost of resource use by automatically adjusting the virtual nodes depending on the real-time data load. A cost-effective auto-scaling (CEAS) framework is first proposed for an Amazon Web Services (AWS) Cloud environment. The proposed CEAS framework allows us to scale the computing resources of Hadoop cluster so as to either reduce the computing resource use when the workload is low or scale-up the computing resources to speed up the data processing and analysis within an adequate time. To validate the effectiveness of the proposed framework, a case study with real-time sentiment analysis on the universities' tweets is provided to analyse the reviews/tweets of the people posted on social media. Such a dynamic scaling method offers a reference to improving the Twitter data analysis in a more cost-effective and flexible way.

Keywords: big data; cloud computing; Apache Hadoop; Amazon web service; Twitter

1. Introduction

Cloud computing has been deployed worldwide due to its versatility and flexibility for storing and transferring data, while maintaining secrecy over three basic layers including Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [1–3]. Cloud computing manages the shifting computing requirements by offering more flexibility in the services provided to organisations. Specifically, virtualisation plays a major role in cloud computing; for instance, distributed computing and delivery of computer services over the Internet [4,5]. There exist a variety of service providers, such as Microsoft Azure, Google Cloud Platform, Oracle Cloud Infrastructure and Amazon Web Services (AWS) [6], which have common elements to support instant provisioning, self-service, auto-scaling and security.

As a typical cloud platform, AWS has provided a large range of services across computing, database, storage, analytics, networking, developer tools, Internet of Things (IoT), security and enterprise applications. The AWS also offers a Virtual Private Cloud (VPC), which logically isolates the AWS space so that one can have a complete control over the virtual environment. Moreover, the use of the AWS resources has been shown to be cost-effective compared to other cloud platforms providing the same services [7]; for instance, Amazon Elastic MapReduce (EMR) which can process big data across a Hadoop cluster of virtual resources on Amazon Elastic Cloud Compute (EC2) instances [8].

Specifically, MapReduce is a framework on which we can write applications to help process an enormous amount of data on a cluster of commodity hardware [9,10], while Apache Hadoop is well known as an open-source software service that enables a large distribution process across different clustering systems using simple programming models [11,12].

Recently, a cloud-based infrastructure has been shown to be much more versatile than the traditional infrastructure from both service and security perspectives, especially when dealing with big data. However, selecting a cloud infrastructure for auto-scaling and optimising the virtual resources for such rapid data growth is a challenging task. Determining scaling policies on new virtual machines as well as setting up a threshold that can automatically configure the computing resources on a Hadoop cluster have therefore attracted more research interests [12–15].

In this paper, dealing with the scalability of cloud computing, we first propose an auto-scaling framework via MapReduce cluster by employing Hadoop on the AWS platform. Depending on cluster load, the proposed framework can scale within itself and the cluster can be monitored and debugged in real-time mode to find appropriate auto-scaling strategies. Furthermore, in order to validate the effectiveness of the proposed approach, real-time sentiment analysis is provided to analyse the Twitter data of various universities. Consequently, the main contributions of this paper can be summarised as follows:

- *A novel cost-effective auto-scaling (CEAS) framework for cloud computing*: An infrastructure on AWS is proposed to perform auto-scaling and efficient sentiment analysis of big data. The clusters can be created of any size through either user interface or code. In the proposed CEAS framework, an Amazon EMR cloud computing platform is developed which enables data processing along with data clustering. The proposed CEAS aims to minimise the cost of resource use by automatically adjusting the virtual nodes depending on the real-time data load. The EMR clusters can be monitored and provisioned by using AWS monitoring services, such as cloud watch or through a dashboard, which allow us to not only visualise and assess the Hadoop cluster performance in real-time, but also help debug for troubleshooting. Moreover, an open-source tool called Ganglia is used to generate the reports on the performance of the Hadoop clusters.
- *An efficient scaling policy for CEAS framework*: Taking into account dynamic workload, a scaling policy is created for the proposed CEAS framework that can either add or remove the clusters so as to optimise the virtual instances. New data storage systems are introduced to store the data while performing the scale-down operations on the dynamic virtual clusters. The resources can be easily either scaled up during heavy workload or scaled down for saving power consumption subject to limited available resources. The novelty of the proposed auto-scaling method is that an absolute amount of resources can be determined and allocated for the AWS to process and analyse the big data.
- *Sentiment Analysis of Twitter data with MapReduce over CEAS framework*: MapReduce is exploited for distributed computation of large data sets over Hadoop clusters of computing resources. Specifically, the MapReduce technique is applied to the auto-scaling cluster via the proposed CEAS framework that allows us to perform sentiment analysis on Twitter data of various universities. The experimental results show that the proposed scaling method in the CEAS framework handles the workload efficiently by dynamically adjusting the virtual computing resources.

The rest of this paper is organised as follows: Section 2 summarises related works on distributed scaling of Hadoop clusters for cloud computing and big data analysis with MapReduce and Amazon EMR as motivation for our work. Section 3 presents the proposed CEAS framework with various functionalities for auto-scaling and sentiment analysis using MapReduce technique. Sentiment analysis of Twitter data and the performance of CEAS framework is presented in Section 4 to validate the effectiveness of the proposed approach. Finally, Section 5 concludes this paper with recommendation for further research.

2. Related Works

A system is scalable if it can be adapted to handle the increasing load when new hardware is added [4,16]. The computing scalability of a system is thus defined as the capability of the system to accommodate such growth of the workflow. Auto-scaling mechanisms are accordingly brought into concern of industrial cloud suppliers with their open supply cloud platforms, such as Eucalyptus, Aneka, Flexiscale, GoGrid, Engine, Google App, Windows and Amazon. These cloud platforms provide several benefits to the cloud computing scalability, cloud security, implementation and management. However, there exist some drawbacks at most of the third parties where the lack of good and consistent Internet connectivity causes a fluctuating cost increase in the hardware configuration every month.

Dealing with a large volume of data, several studies revealed the use of Hadoop for processing big data. It is vital that such large number of datasets should be adjusted to handle the dynamic process of workload through the existing computer resources [12,14,15]. A Hadoop cluster with a fixed size is shown to have problems when processing the streaming data and it is also not cost-effective for the cloud resources since it does not use fully its resources when the data load on the cluster is low. In [13], Leverich and Kozyrakis introduced a method to scale down the cluster to improve the resource use and energy consumption for cost efficiency. In the proposed method, the removal of the cluster nodes was investigated, but the operation related to the scaling process was hardly considered. Also, when the workload exceeds, the capacity of computing resources will be degraded. Aiming to optimise the Hadoop cluster, Maheshwari et al. proposed in [17] a Hadoop cluster reconfiguration and data replacement to either shut or open the instances by checking the resource use rate. However, before removing the node, the stored data on one instance has to be transferred to another node, which may take a long time for extensive data. The scaling issue was also not fully managed when adding machines would even cause a higher cost given limited physical machines and infrastructure.

Cloud computing has emerged as a solution for scaling instances by using clusters as web services [1,3,4]. In [18], a customised load balancing and a distributed dynamic algorithm were proposed for auto-scaling of virtual clusters with a load balancer which can handle the requests sent from users for the EC2 instances in a dynamic manner. The EC2 virtual instances can be either increased or decreased as per-the-CPU use and with respect to incoming user requests on the virtual nodes, whereas the load balancer can distribute the incoming data traffic automatically on the EC2 instances, containers and network IP addresses. Although the proposed load balancer offers automatic scaling, robust security and high availability for fault-tolerant applications, it does not support Hadoop for processing large volume of data. In [12], MapReduce was proposed as a solution for big data analysis with Apache Hadoop over Hadoop Distributed File System (HDFS). Specifically, minimisation technique was applied for indexing along with shuffling, sorting, mapping and reducing the files. The MapReduce supports distributed and parallel input/output scheduling which is tolerant to failures and thus promote the scalability of cloud with large dataset, especially when considering dynamic processing workload. Integrated with MapReduce, Apache Hadoop has been designed to be able to scale up from a single system to multiple systems [12], in which MapReduce is employed as a programming model for processing and distributing java based computing. Basically, MapReduce contains two major tasks including [9,10]: (i) *Map stage*: converting a data set into some other data sets where individual elements are referred as keys or value pairs; and (ii) *Reduce stage*: Fetch the results from the Map stage, which are then transformed into smaller sets. Such two-stage process facilitates the scaling of the data processing over multiple computing nodes on the Hadoop platform.

MapReduce was shown to be of benefit to large-scale data-intensive applications with several iterative computations, e.g., in [19] where Twister4Azure was developed as a distributed decentralised iterative MapReduce for Microsoft Azure cloud computing platform. Hadoop was also adopted in [20] for processing big geospatial data in the cloud. An auto-scaling framework was developed and evaluated via a prototype system employing digital elevation model interpolation of the collected geospatial data for GIScience applications. Another Hadoop framework, namely Virtual Hadoop,

was proposed in [21] using Docker containers to facilitate the auto-scaling mechanism in heterogeneous computing environment. Such dynamic scaling in Hadoop however has an inherent trade-off between the computation power with additional nodes and overheads for data redistribution [22].

Deploying and managing Hadoop clusters are however time-consuming and challenging over the traditional underlying server. Instead, Amazon EMR can facilitate this process by exploiting the elastic infrastructure of EC2 and Amazon Simple Storage Service (S3). The Amazon EMR offers a manageable Hadoop framework that distributes the computation of data over multiple Amazon EC2 instances along with the Amazon S3 cloud-storage services which provides a system interface to decrease the complexity of the traditional hardware management [7,8]. With the Amazon S3, the impact of the quality of service experienced by the users depends on the location and the configuration of the services. Specifically, the cloud storage is aimed to improve the knowledge of cloud-to-user network experience based on the Amazon S3 and the customer data can be organised using the objects stored in the buckets which are placed in the cloud region.

To handle a huge amount of unstructured and structured information in real time, Azure MapReduce was proposed in [23] to cope with big data issues. Azure has several services like Azure Queues which are used to schedule tasks and Azure blob storage for storing data. Moreover, it enables the implementation of auto-scaling Hadoop cluster which leads to a higher level of flexibility and fault tolerance. However, this approach is not as flexible as AWS and is also not cost-effective [9]. Additionally, in the traditional Hadoop cluster with a First-In First-Out (FIFO) scheduler, all task slots for a job in the cluster blocks other jobs to use the resources until the current scheduled job finishes, which may cause a bottleneck in processing the data [14], especially when dealing with massive data (e.g., Twitter) arriving at the same time. As one of the biggest social platforms, Twitter provides a huge data set which can be streamed for sentiment analysis. An example of Twitter data analysis can be found in [24] where a prototype was developed for text mining by combining Twitter streams, Python text processing and Cassandra storage methods. Dealing with the computation of such increased data over social media, it is required to have an enhanced framework that does not rely on hard disk storage, but can process data in memory instead [23].

Despite the amount of research done to date in the cloud-based applications and services, little of this research is beneficial in terms of dealing with elasticity. To this extent, this paper develops a new cloud platform with cluster configuration and scaling policies to deal with auto-scaling and data analysis. Moreover, we introduce some new ideas to strengthen the network provisioning, availability, security and scalability towards providing an efficient, reliable and manageable resource for cloud computing.

3. Proposed Cost-Effective Auto-Scaling (CEAS) Framework

The primary objective of the scaling method is to improvise the scalability for cloud computing. In this section, the conventional auto-scaling framework for cloud computing in Hadoop system is presented outlining key components that enable the scalability of virtual resources on the cloud.

To process a large volume of data efficiently in real time, it is crucial to improvise auto-scaling and sentiment analysis of the big data on the cloud. This section initially presents our proposed CEAS framework for collecting and aggregating data on the AWS platform. The configuration used for setting up the EMR clusters for efficient data processing is then discussed in detail along with our proposed solution for implementing the CEAS framework for sentiment analysis of Twitter data based on cost effectiveness, high throughput and high reliability on the cloud environment.

Figure 1 illustrates the proposed CEAS framework which consists of the following six components:

(i) *Cloud computing platform*: The cloud computing offers scalability of virtual resources on either public or private clouds (e.g., Amazon EC2, Amazon EMR, Microsoft Azure, OpenStack and Eucalyptus).

(ii) *Covering HDFS-enabled Hadoop cluster*: Covering HDFS of Hadoop cluster on the EMR cloud has provisioned computing resources to be able to manage, store and process a large volume of data.

(iii) *Cluster monitor (or cloud watch)*: Monitoring clusters plays a vital role in auto-scaling framework, which allows us to capture the information about all the resources on the cluster by using

Application Programming Interface (API) of Hadoop, such as CPU use, the performance of map and reduce tasks, and the status of the jobs.

(iv) *Auto-scaler*: The auto-scaler provisions the virtual resources on the cloud platform using cloud's API (e.g., Amazon EMR API) which is dependent on the information gathered by the cloud watch. When the data on the cluster increases, more nodes are added to the Hadoop cluster, while the idle nodes are shut down when the data load is less.

(v) *Ganglia*: Ganglia is a front-end open-source monitoring system which is scalable and distributed to monitor the clusters or grids. This monitoring system offers a complete solution for archiving, monitoring and visualisation of the system performance.

(vi) *Sentiment analysis*: The sentiment analysis is performed on the auto-scaling cluster by taking the data from multiple universities via Twitter app.

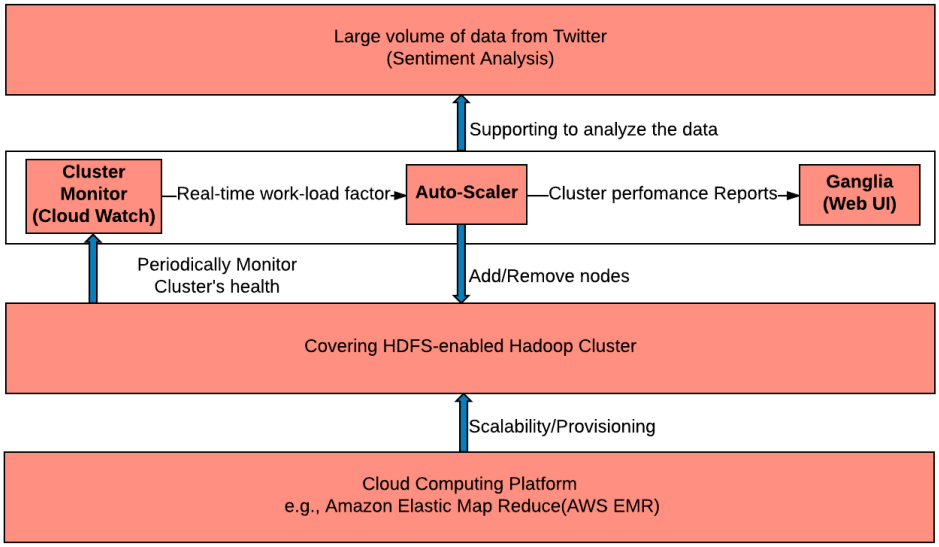

Figure 1. Cost-Effective Auto-scaling (CEAS) framework.

In the CEAS framework, the above-mentioned Hadoop's auto-scaling framework is adopted on the AWS cloud to automatically scale the Hadoop clusters so as to allocate the Amazon Machine Images (AMIs) (The AMIs provide the information required to launch EC2 instances, which includes the volume of instances and permissions that control these instances.) based on the dynamic processing workload.

Amazon AWS allows for multiple scaling groups within a scaling plan that is based on optimisation for availability, cost or balance of both. However, the scaling plan would be enabled for all scalable targets, instead of individual targets." Thus, based on the scaling options available in AWS, in the scope of this work, an auto-scaling framework is defined as effective if it meets the following requirements:

(R1) A cloud environment should be cost-effective, while still being able to perform auto-scaling and sentiment analysis. The framework also supports deployment, configuration and management of Hadoop clusters.

(R2) A cloud platform should be able to monitor all tasks in an efficient way for all workload in real time. Moreover, when the task is completed, it would be able to shut down the cluster. Also, it should provide elasticity for the cloud computing by either expanding or shrinking the virtual cluster to handle the data.

(R3) An efficient sentiment analysis should be performed via the auto-scaling cluster with low cost and short scaling time.

In Figure 1, the enhanced functions for the proposed CEAS framework to fulfil the above three requirements will be described in the following subsections.

3.1. Amazon EMR Cloud Computing Platform

In the proposed CEAS framework, AWS cloud is selected to facilitate the Hadoop applications for processing big data on a cloud environment. Specifically, Amazon EMR is exploited for scaling Hadoop clusters due to its benefits over other cloud environments, e.g., in [25], which can be summarised as follows:

(i) *Cost saving*: The cost of EMR relies on the number of EC2 instances, the instance type and the region where the cluster is launched. On-demand pricing provides a lesser rate per hour, but the cost can be brought down by purchasing Reserved or Spot instances.
(ii) *Service integration*: The EMR can work together with other AWS services (e.g., Amazon EC2 instance, Amazon Virtual Private Cloud, Amazon S3, Amazon Cloud Watch) to offer networking, security and storage for the Hadoop cluster.
(iii) *Compatibility*: The EMR allows us to configure other applications like Hadoop, Hive and Spark on EC2 instances.
(iv) *Scalability*: The EMR provides the flexibility to increase or decrease the cluster size by checking the data on the Hadoop cluster.
(v) *Reliability*: The EMR watches over the EC2 instances running in the cluster, and thus can replace or terminate the nodes automatically if there is a failure.
(vi) *Security*: The EMR provides the security for the Hadoop clusters and data by using EC2 key pairs.

In the AWS cloud under investigation, the EMR offers a manageable Hadoop framework that distributes the data computation over multiple virtual Amazon EC2 instances and loads the data processing applications into Amazon S3. In this EMR architecture, the data can be stored in the Amazon S3 with respect to its size, compression and/or partitioning. Prior to processing the data, it is only required to be copied to HDFS once and does not need to be copied multiple times from Amazon S3 to Amazon EMR cluster due to the fact that the data processing task performs iterative work on a single data set. Thus, considerable reduction in processing times are achieved.

In particular, this architecture is shown to be beneficial for scaling when the Amazon S3's scale allows the EMR nodes to run as many as needed to pull the data in parallel and the HDFS nodes can scale up when the data size increases. This implies that the requirement for additional storage of the HDFS nodes can be eliminated by exploring the storage capacity of the Amazon S3. Additionally, it is safe to shutdown the Amazon EMR cluster once the data processing job is done, while multiple jobs can be triggered on the same data set in the cluster without overburdening the nodes in the HDFS.

3.2. Data Processing and Clustering with Amazon EMR

The Amazon EMR enables the processing of large data and the workflow of the data processing can benefit from virtual Hadoop clusters [15,25]. When the Amazon EMR cluster is provisioned, it is crucial to choose the optimal number of instances and their sizes since certain workloads may be disk-I/O or memory intensive when the others are CPU intensive.

In the proposed CEAS framework, for comparison between the fixed cluster and auto-scaling cluster, four kinds of Hadoop cluster are considered in the AWS cloud environment with the following settings:

(i) *Three-slave cluster*: 3 slaves with 3 large instances;
(ii) *Seven-slave cluster*: 7 slaves with 7 medium instances;
(iii) *Fourteen-slave cluster*: 14 slaves with 14 medium instances;
(iv) *Auto-scaling cluster*: a different number of slaves which varies from 3 to 15 slaves with the corresponding instances.

Intuitively, the fourteen-slave cluster should provide the best performance when processing the big data. However, such employment of all available slaves may be not necessary when the Hadoop only uses the memory as per the requirement of data processing. Also, considering the scenario when memory is not sufficient enough to perform the sorting on large data sets, if certain part of the data is written to disk which is expensive and may slow down the job, then using the instances with high memory is preferred to achieve a better performance. With the Amazon EMR, various instance sizes can be implemented by either switching among various instance profiles or terminating the current cluster and starting a new cluster with different instance size.

The demands for both CPU and memory-intensive jobs can be further met by mixing various types of Amazon EC2 instances and by adding or removing the cluster nodes. When provisioning the cluster, depending on the size of the instance, the Amazon EMR can set up various optional settings for the Hadoop configuration, such as the size of the Java memory used per daemon, and the number of mappers and reducers configured per instance.

To quickly process the data on Hadoop cluster, the required data nodes need to be added to the cluster based on the data size. Hadoop can process the data by splitting the input files into blocks for parallel processing. Moreover, Hadoop can create more data blocks and tasks to analyse the massive data. The opted Amazon EMR cloud can therefore have only one cluster but with several nodes that can analyse and process the map reducing jobs in parallel. For instance, if a dataset needs 30 mappers to process the input data file consisting of 30 splits realised by Hadoop, then an ideal Amazon EMR cluster would be capable to manage all mappers running in parallel by employing ten small instances each of which can execute three map tasks concurrently.

Although parallel data processing is shown to be advantageous, the usage of all mappers is actually not necessary when there is no time constraint and the excessive cost for setting EMR Hadoop cluster is also a major concern. Instead, we can reduce the number of mappers for latency-tolerant jobs. However, the data stream on the HDFS system is distributed across the slave nodes, and thus the scale-down operation on the clusters may switch off a node causing the data loss [11,12,25]. In Hadoop, decommissioning was developed to avoid such data loss by transferring the data from an instance to other machines prior to removing that instance. The time consumption for such decommissioning depends on the data size, but this is not tolerable to perform an adequate scaling. Hence, auto-scaling has been proposed to timely collect the streaming data load. By using the Covering HDFS in Hadoop, the cluster can scale down the instances without suffering any data loss [13].

A typical Hadoop cluster consists of [20]: (i) *Core slaves*: to provide both computational power and storage services, and (ii) *Compute/task slaves*: to offer only computational power services. By employing slaves with different tasks, the Hadoop clusters can be automatically provisioned with the required data nodes in the AWS cloud.

3.3. Monitoring and Provisioning Amazon EMR Clusters for Auto-Scaling

To monitor Amazon EMR clusters, as illustrated in Figure 1, a cloud watch or a cluster monitor is required in an auto-scaling framework, and thus should also be employed on AWS platform of the proposed CEAS framework to continuously track and monitor all resource information including the status of the jobs and map reducing tasks. In this subsection, elastic Amazon EMR cluster is first described along with the requirement of monitoring and provisioning services, applications and resource use. The rules and implementation of auto-scaling are then presented for the proposed CEAS framework.

3.3.1. Elastic Amazon EMR Cluster

As discussed in the previous subsection, an Amazon EMR cluster which is made of Amazon EC2 instances has sufficient instances to process the streaming data. Each EC2 instance in the cluster is classified as a node and the cluster can monitor the requirements as the data size increases. The cluster

also coordinates the resizing of the cluster to suit everyday data request needs [15,25]. There are basically three types of nodes available within the Amazon EMR cluster, including:

1. *Master node*: executes NameNode and JobTracker, coordinates data distribution and tasks among other nodes, while monitoring the cluster health.
2. *Core/Slave nodes*: executes DataNodes and TaskTracker, runs the tasks and stores data in the HDFS.
3. *Task nodes*: are optional nodes added to handle the TaskTracker when a high load is required for data processing.

On AWS cloud, CloudWatch offers a periodic monitoring service for applications and resource use at every 5 min. The EMR CloudWatch metrics can be used to automate the task by adding the required data nodes to the cluster. The Amazon EMR uses a variety of CloudWatch metrics like the number of mappers and/or reducers (active/inactive), the status of the cluster (idle/active), and HDFS use to resize the cluster.

Using Amazon Simple Notification Service (SNS) along with CloudWatch metrics, a notification service can be enabled for resource use within an EMR cluster. For instance, the EMR cluster with a large number of jobs during peak hours needs to be resized rather than increasing the queuing time of the jobs. Such process can be monitored using CloudWatch metrics and auto-scaling action could be performed. By using the Amazon EMR, the processing time can be controlled and further improved by adding the required capacity. Moreover, the CloudWatch can trigger an alarm when CPU use increases. This alarm can be associated with an auto-scaling policy via SNS notifications, i.e., a scaling event is logged when the scaling is triggered.

3.3.2. Rules for Auto-Scaling

For an instance group, when a scaling activity is triggered by a scale-out, Amazon EC2 instances are added to the instance group based on the scaling rules, whereas the termination and removal of instances can be done by setting up a scale-in rule. With simple scaling policies available with Amazon AWS, the threshold values for the CloudWatch alarms that trigger the scaling process as well as define how the Auto Scaling group should be scaled when a threshold is in breach for a specified number of evaluation periods. In our framework, we have chosen the metric *PercentChangeInCapacity*, where we specify the minimum number of instances to scale (using the MinAdjustmentMagnitude parameter, Add instances in increments). The autoscaling policy is based on the two common metrics of CloudWatch—*YarmMemoryAvailablePercentage* and *ContainerPendingRatio* provided within the AWS.

The scaling behaviour for each rule in a scaling policy is identified by the following parameters:

- *Scale-out and scale-in rules*: apply to the maximum and minimum instances, respectively.
- *Unique rule name*: should be configured in the scaling policy.
- *Resizing of EC2 instances*: must be determined by the scaling adjustment.
- *CloudWatch*: monitors for an alarm threshold condition and compares the performance metric against a threshold value.
- *Evaluation period*: is the duration when CloudWatch metric is evaluated and in triggered condition before prompting the auto-scaling activity.
- *Cooldown period*: is the time passed by the instance between the period when the auto-scaling activity is triggered by the scaling rule and the beginning of the next scaling activity.
- *Performance metrics*: are variables that need to be monitored, such as CPU usage, traffic, etc.
- *Cloud watch alarm*: is an object that controls a single metric over a given period. An example of a threshold value to create an alarm is shown in Figure 2. The scaling of resources can be thus triggered within the EMR.
- *Alarm*: is used to give warning when the value of the EMR metric breaches a defined range and maintains the change for a specified duration.

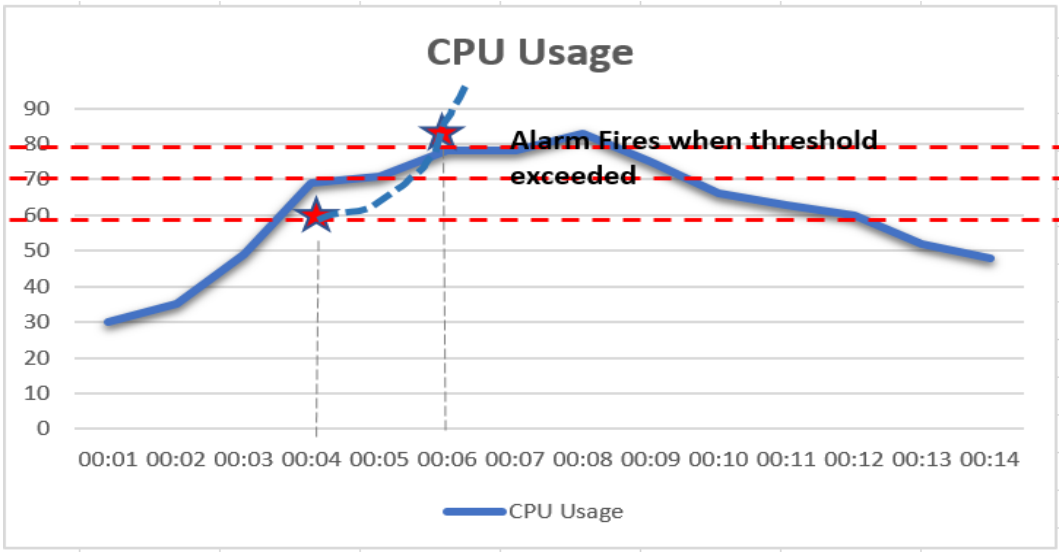

Figure 2. Alarm threshold setting based on CPU usage.

Within each Hadoop cluster, by enabling the Ganglia web interface, the performance of the entire Hadoop cluster can be evaluated and displayed by visual reports. Also, the performance of each individual instance can be inspected. Here, Ganglia is referred to as an open-source tool for high-performance monitoring of grids and clusters. It consists of gmond, gmetad and a front-end web. Gmond is a multi-threaded daemon that runs over all nodes and information is reported back to gmetad which helps in collecting the metrics and also generating the report. In the CEAS framework, the Ganglia is initially used for graphing and data logging. The data is visualized by the front-end web bundled with Ganglia which offers a complete solution for archiving, monitoring and visualisation of the system metrics without any special coding.

3.3.3. Auto-Scaling EMR Clusters

This subsection details how the proposed CEAS framework saves processing time for the jobs based on real-time workloads in the AWS.

To process jobs or to interact with the software deployed over the EMR cluster, a secure connection must be established to the master node, then the interfaces and tools can be accessed. First, the input data is processed by using the MapReduce utility on the Hadoop cluster. The data is then stored in an HDFS file system or Amazon S3. Finally, the stored data is passed for processing in a sequential manner and the processed output is written to a specific HDFS or S3 location.

To successfully run the allocated jobs, Amazon EMR cluster follows the following two steps:

- *Step 1 (STARTING)*: The applications deployed on a Hadoop cluster is provisioned by using EMR. The cluster state is at STARTING point.
- *Step 2 (BOOTSTRAPPING)*: Additional applications, if required, can be installed on the cluster. This is known as BOOTSTRAPPING process.
- *Step 3 (RUNNING)*: The cluster runs all the steps sequentially after the above two steps have completed. The cluster state is thus called RUNNING.

Upon completion, the cluster goes to a wait or shutdown state. In case of a cluster failure, it terminates all the EC2 nodes and deletes the data saved on the slave nodes.

4. Performance Evaluation of the Proposed Framework

The effectiveness of the proposed approach is validated through extensive simulations and a case study using real-time sentiment analysis on Twitter data of several institutions. The performance evaluation is done in terms of scalability and capacity of the Hadoop cluster and the average time

taken for all the real-time workloads to complete on the computing resources. A comparison is provided between the designed static and auto-scaling cluster of the relaying Hadoop schemes, such as MapReduce and Ganglia. The four Hadoop cluster setups as previously described are considered: (1) three-slave cluster, (2) seven-slave cluster, (3) fourteen-slave cluster, and (4) auto-scaling cluster. The time consumed by each map reducing jobs on the created Hadoop clusters are recorder over one-hour and three-hour time frames.

4.1. Test Case of the Proposed CEAS Framework

This subsection presents a test case of the proposed CEAS framework for sentiment analysis on Twitter data using MapReduce utility in Hadoop clusters.

Opinion mining is a well-known service to orchestrate what users post on different social media platforms. Many users share their views on different aspects of life every day. Efficient techniques should be developed to collect a significant amount of social media data and extract meaningful information from them [26]. To investigate the CEAS framework, Twitter data is taken as it has the ability for users to tweet about events, situations, feelings, opinions in real time. A massive amount of Twitter data is accumulated from various Twitter sources by using Twitter API for sentiment analysis of Big Data [23,27].

This work aims to provide an automatic scaling system which predicts the sentiment of the tweets of the people posted on social media. Hadoop clusters are employed to process the data and MapReduce programming is used to analyze the sentiment statistics on the Twitter dataset due to its capability in performing distributed and parallel processing on large data sets [10,14]. The data from different universities in the United Kingdom is collected by following a Twitter API link to perform sentiment analysis. It is worth noticing that the analysis of the universities' Twitter data in the proposed CEAS framework has benefits over those based on the previous data sets in the sense that the tweets are now analysed in real-time.

4.2. Cluster Setup Performance

Figures 3 and 4 illustrate the time consumed by various cluster setups over a one hour and three hour time periods, respectively, for processing 20 GB of data. The time consumed represents the aggregated time consumption of each workload. It can be noticed in Figure 4 that the seven-slave cluster setup on the AWS cloud takes much longer to process the data, when compared to the fourteen-slave cluster or the auto-scaling cluster setups. However, this situation happens for the heavy data loads, and this is demonstrated by the increased time consumed starting at minute 91 and decreasing by minute 156 as seen in the graph. While the data load on the cluster is exceeded, the processing jobs of input data queue in a job tracker waiting to be assigned when the old jobs are completed. More precisely, submitted jobs in a job queue wait a long period of time when there is an increase in the data load.

The fixed cluster setups of three, seven and fourteen slave nodes therefore restrict capacity to process the map reducing jobs on the streaming data load on the Hadoop cluster. Moreover, the results show that the time consumption of each workload for the auto-scaling cluster is almost the same as the fourteen-slave cluster setup. However, it can be noticed in Figure 4 that the auto-scaling cluster presents some fluctuations starting at 1.5 h. This is not the case for the fourteen-slaves cluster setup. The sharper fluctuations indicate the scaling-up of the resources on the cluster in order to finish the jobs on time. The scale-up performance involves adding task/slave nodes on the cluster to make sure that the jobs complete on time.

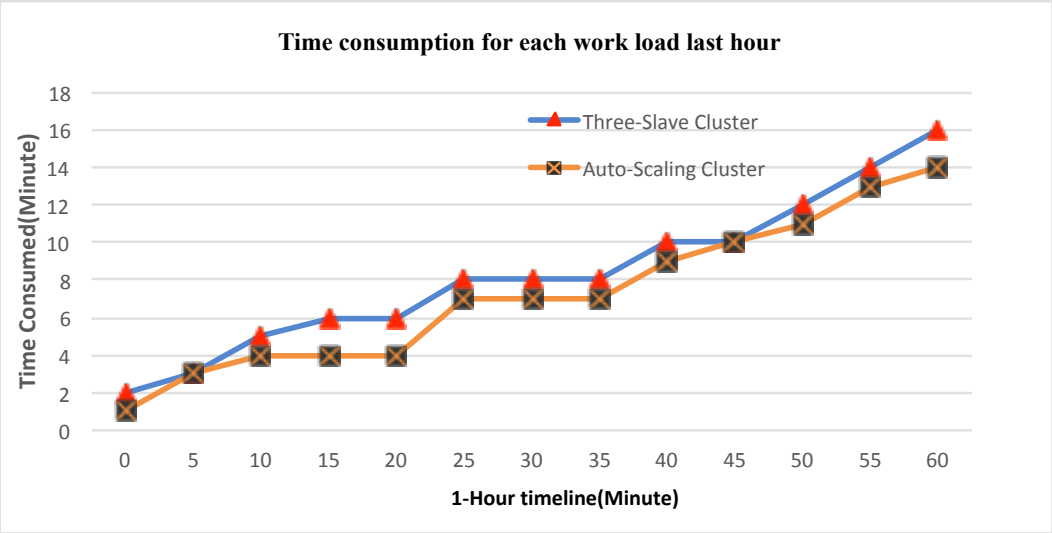

Figure 3. Time consumption for each workload last hour.

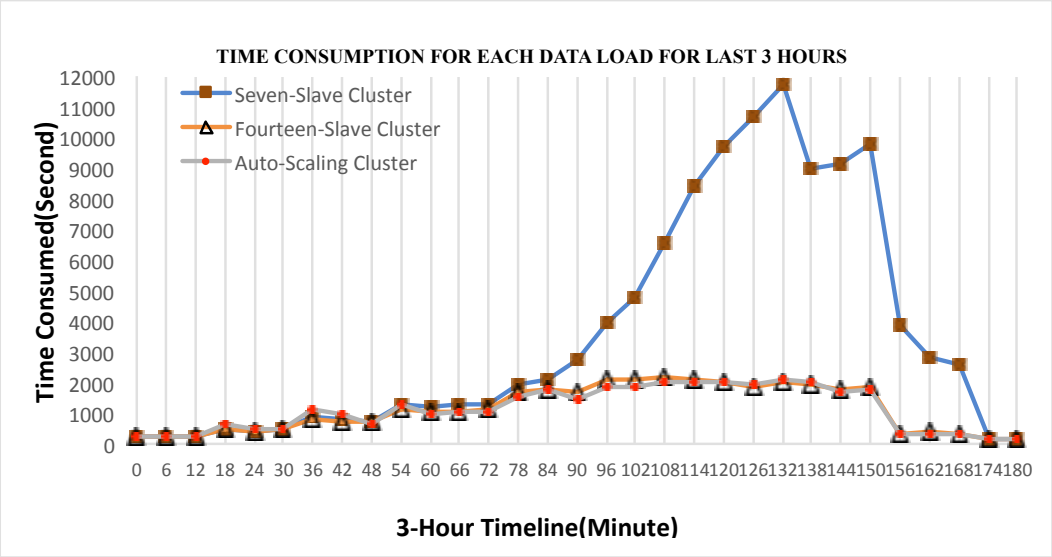

Figure 4. Time consumption for each workload last three hours.

4.3. Resources Utilisation and Data Locality

During the peak workload, fixed and auto-scaling clusters do not have idle slaves. When the workload falls back, the fixed clusters have more idle slave nodes than auto-scaling cluster. To process and efficiently analyse big data, the MapReduce jobs can be carried out on both the Hadoop fixed and auto-scaling clusters. Figure 5 illustrates the aggregated time consumed to accomplish all the workloads. In the auto-scaling cluster, there are always a minimum of 2 and a maximum of 14 machines available depending upon the workload. The results show that for the same MapReduce job when running on the auto-scaling cluster compared to the fixed fourteen-slaves cluster, takes 9.6 h and 9.72 h to finish respectively. This accordingly demonstrates that the auto-scaling cluster processes and finishes the jobs in an efficient manner and thus saving not only 12 min, but also number of instances.

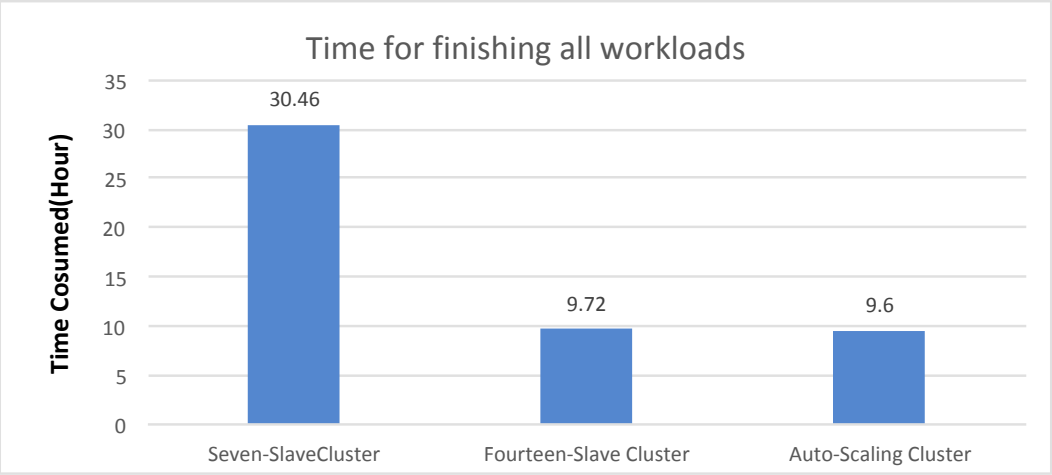

Figure 5. Time consumption for all workload.

4.4. Amazon On-Demand EC2 Instances Pricing

Table 1 indicates the Amazon EC2 instances pricing. These charges are only applicable for the used EC2 instances. Moreover, the main use of on-demand EC2 instances frees the purchasing, complexities of planning and hardware maintenance and it transforms the large fixed costs into smaller costs. Based on the pricing given in Table 1, the charges calculated for the auto-scale cluster, which scaled up to 12 machines and scaled down to 2 machines, summed up to $27 to complete the jobs for the heavy workload. Moreover, the charges applied to the fixed fourteen-slaves cluster was $33 to complete the jobs with a heavy workload. Whereas, the cost of the used resources is higher than the resources used in the Auto-scaling cluster. Thus, the auto-scaling cluster saves more instances and time when the created cluster is supposed to run for two months with constant data loads, which is equivalent to saving the amount of $419/month and $838/2months when compared to the fixed clusters.

Table 1. Amazon EC2 instances pricing.

EC2 Instances	CPU Cores	Memory (GB)	Unix/Linux Usage
t2.medium	2	4	$0.05 per Hour
c3.xlarge	4	7.5	$0.239 per Hour

4.5. Data Locality in HDFS Cluster

Within the HDFS cluster, the core nodes work as input or output channels for the created clusters, and it also provides data storage. The tasks performed by the task tracker play a vital role on the core nodes taking advantages of the data storage. Core slaves on the physical cluster have more capacity and storage to perform the scalability. Moreover, core instances cannot be terminated even they are idle. As per the above results, the scaling cluster used the private cloud as AWS and scaled based on the Covering HDFS methods, whereas the fixed clusters use the traditional infrastructure. As displayed in Figures 3 and 4, the proposed Covering HDFS in the AWS does not cause any issues without any performance degradation. The auto-scaling methods are more flexible in balancing the cost and cluster's performance. In some cases, applications use a small amount of core nodes to avoid the cost of the resources when the data is low. To achieve the best performance, the data nodes can be added when the data is high. The fourteen-slave cluster setup on the traditional Hadoop infrastructure performs well with the big data but it wastes more resources when it has less data to process. The seven-slave cluster in Figure 4 displays the best resource use rate when compared to the fourteen-slave cluster. Also it can be noticed that the performance is degraded in the seven-slave cluster in case of heavy data load.

Overall, the auto-scaling cluster showed the best performance vs. cost trade-off and better resource use by dynamically adjusting the virtual computing resources on the Hadoop cluster.

4.6. Sentiment Analysis of Big Data

This subsection presents a use-case scenario of real-time sentiment analysis on Twitter data of several universities collected using Twitter app. The main idea is to analyse thousands of tweets coming at every second within a very short amount of time. The framework should be dependent on the imported data volume. This is important because the volume of tweets is growing at a noticeable rate and the cluster size needs to scale up based on the data size.

In this real-time study, the Twitter data was collected for sentiment analysis in the form of a live stream, from eight UK Universities, including: Brunel University, Imperial College London, Kingston University, Queen Mary University of London, Liverpool University, Bedfordshire University, Middlesex University and Cambridge University. To analyse the streaming data at a high data rate, the data is processed using the proposed CEAS framework with auto-scaling cluster on Hadoop and MapReduce is employed to perform the sentiment analysis on the universities' Twitter data.

Over 200 million twitter messages collected from the above mentioned universities are used to identify and classify the feedback given to the universities. Twitter's timeline APIs are exploited to retrieve the universities' data. Here, the tweets contain the information about that specific Tweet and the name of the user who tweeted on the social web. Specifically, the raw json tweet mainly contains the following fields: (1) Identification of the Tweet by Tweet ID, (2) user names associated with Tweets, and (3) the text of the tweet and its timestamp.

The Tweets are classified into: *positives*, *negatives* and *neutral tweets*. As a means to perceive and differentiate user opinions, the sentiments should be detected in the post. Examples are given below for positive and negative tweets with a notice that they may reflect completely different opinions:

- *Example 1*: "text":"@georgierdarling Hi @georgierdarling I would be happy to help! I am a voice coach and senior lecturer in voice at Brunel Uni
- *Example 2*: "text":"RT @Shan_kelly96: It is extremely disappointing to see that Brunel university have members of staff working as the voice of the students and collected tweets from Rest API by using the Twitter4j library.

There exist many challenges in the sentiment analysis. For instance, in the above examples, the user is expressing his own interest towards his role in the first example. However, an opinion word which is considered to be positive in one state may be seen as negative in a different situation. This is due to the fact that people may not always express the same views. This indeed can be noticed in the second example in which the user's opinion is different and tweeted against the first one. In general, both information can be used even if these are contradicting each other.

The sentiment analysis of Twitter data is realised as follows: in the preprocessing phase, the tweets are available as raw text data. The punctuation and other symbols can be removed from the tweets as it might affect the accuracy of the overall evaluation process. The tweets are labeled based on University names. This data is then processed using the proposed CEAS approach with auto-scaling cluster configuration of the Hadoop framework as follows: At first, Mapper takes tweets from the universities json and calculates the weights on each tweet by using the weights dictionary and categorises the university names and weights into the output as a key-value pair. Then, Reducer takes the outputs from the Mapper and aggregates all the tweets for each Mapper's outputs.

Figure 6 shows the aggregate results obtained from the Universities' tweets data. These results can be used to study the user's opinion about the Universities and also to compare these opinions between different Universities which ultimately helps the user in making a good University selection. The aggregated results of the tasks are performed on the auto-scaling Hadoop cluster for the sentiment analysis. To keep track of the jobs, the auto-scale Hadoop framework has three types of counters including:

- *File System Counters* - tracks the bytes read and written by the file system;
- *Job Counters* - keeps track of the jobs and maintains the job level statistics;
- *Map Reduce Task Counters* - collects information about tasks over the execution.

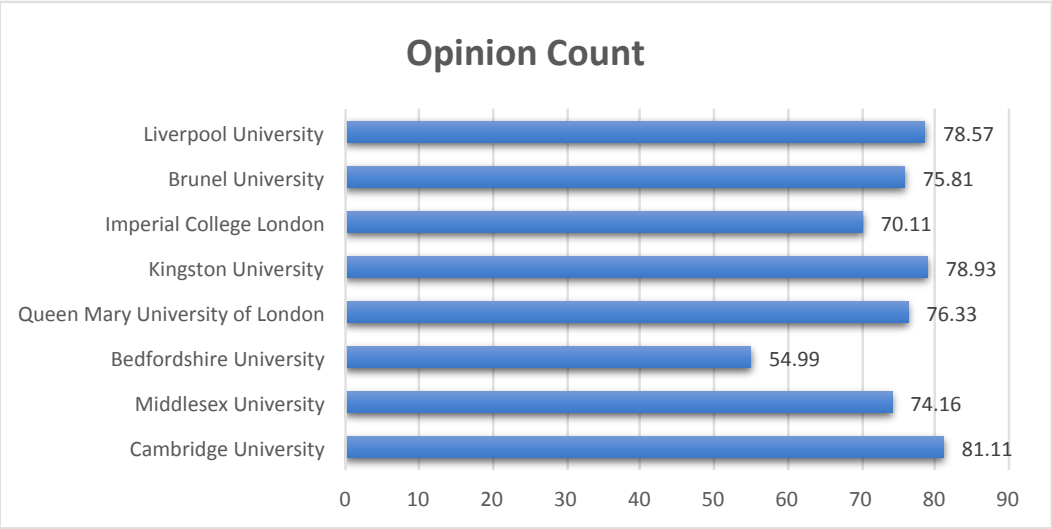

Figure 6. Aggregate results from different Universities.

These results demonstrate the scalability of the auto-scaling Hadoop cluster on the AWS cloud. The sentiment analysis performed on the Universities' Twitter data using MapReduce proves the suitability of the proposed CEAS framework with auto-scaling cluster for big data analytics.

5. Conclusions

This paper has proposed a CEAS framework to automatically scale the cloud computing resources with the aim of improving the cluster's performance and efficiency. The proposed framework with auto-scaling policies and cloud watch metrics allow us to identify the amount of slave nodes to be removed or added. Specifically, the HDFS mechanism on the AWS over computing and core nodes has shown to not only avoid data loss but also save resources. The scalability, efficiency and affordability of the scaling cluster have been demonstrated in the AWS cloud. Moreover, through experiments we have assessed the scaling cluster in terms of resource use rate and scalability by testing the real-time data loads during three-hour and one-hour time frame periods along with the employment of MapReduce applications for sentiment analysis of Universities' twitter data. The employment of the proposed method on Hadoop cluster has facilitated the analysis of user's opinions about different Universities, which accordingly demonstrates the efficiency of the auto-scaling Hadoop cluster on the AWS cloud and its suitability for big data analytics.

Author Contributions: R.J. devised the research problem, performed the simulation and collected data; Q.-T.V., P.S. and R.T. summarised and analysed the data; Rachana Jannapureddy and Q.-T.V. prepared the draft; P.S. and R.T. proofread and finalised the paper. All authors have read and approved the final manuscript.

Funding: This research received no external funding.

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

Abbreviations

AMI	Amazon Machine Image
API	Application Program Interface
AWS	Amazon Web Services
CEAS	Cost-Effective Auto-Scaling
CPU	Central Processing Unit
EC2	Elastic Cloud Compute
EMR	Elastic MapReduce
FIFO	First In First Out
GB	Gigabyte
HDFS	Hadoop Distributed File System
IaaS	Infrastructure as a Service
IP	Internet Protocol
IoT	Internet of Things
I/O	Input/Output
JSON	JavaScript Object Notation
PaaS	Platform as a Service
SaaS	Software as a Service
S3	Simple Storage Service
SNS	Simple Notification Service
VPC	Virtual Private Cloud

References

1. Serrano, N.; Gallardo, G.; Hernantes, J. Infrastructure as a Service and Cloud Technologies. *IEEE Softw.* **2015**, *32*, 30–36. [CrossRef]
2. Curran, K.; Carlin, S. Cloud Computing Security. *Int. J. Ambient Comput. Intell.* **2011**, *3*, 14–19.
3. Bouayad, A.; Blilat, A.; Mejhed, N.E.H.; Ghazi, M.E. Cloud computing: Security challenges. In Proceedings of the 2012 Colloquium in Information Science and Technology, Fez, Morocco, 22–24 October 2012; pp. 26–31.
4. Rittinghouse, J.; Ransome, J. *Cloud Computing: Implementation, Management, and Security*, 1st ed.; CRC Press, Inc.: Boca Raton, FL, USA, 2009.
5. Hwang, K.; Dongarra, J.; Fox, G.C. *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*, 1st ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2011.
6. Bermudez, I.; Traverso, S.; Munafò, M.; Mellia, M. A Distributed Architecture for the Monitoring of Clouds and CDNs: Applications to Amazon AWS. *IEEE Trans. Netw. Serv. Manag.* **2014**, *11*, 516–529. [CrossRef]
7. Tamrakar, K.; Yazidi, A.; Haugerud, H. Cost Efficient Batch Processing in Amazon Cloud with Deadline Awareness. In Proceedings of the 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), Taipei, Taiwan, 27–29 March 2017; pp. 963–971.
8. Ekwe-Ekwe, N.; Barker, A. Location, Location, Location: Exploring Amazon EC2 Spot Instance Pricing Across Geographical Regions. In Proceedings of the 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Washington, DC, USA, 1–4 May 2018; pp. 370–373.
9. Iordache, A.; Morin, C.; Parlavantzas, N.; Feller, E.; Riteau, P. Resilin: Elastic MapReduce over Multiple Clouds. In Proceedings of the 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Delft, The Netherlands, 13–16 May 2013; pp. 261–268.
10. Chalvantzis, N.; Konstantinou, I.; Kozyris, N. BBQ: Elastic MapReduce over Cloud Platforms. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 766–771.
11. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The Hadoop Distributed File System. In Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Lake Tahoe, NV, USA, 6–7 May 2010; pp. 1–10.
12. Manikandan, S.G.; Ravi, S. Big Data Analysis Using Apache Hadoop. In Proceedings of the 2014 International Conference on IT Convergence and Security (ICITCS), Beijing, China, 28–30 October 2014; pp. 1–4.

13. Leverich, J.; Kozyrakis, C. On the Energy (in)Efficiency of Hadoop Clusters. *SIGOPS Oper. Syst. Rev.* **2010**, *44*, 61–65. [CrossRef]
14. Lakshmi, A.S.; BalRaju, M.; Chandra, N.S. Towards optimization of Hadoop Map Reduce jobs on cloud. In Proceedings of the 2016 International Conference on Computing, Analytics and Security Trends (CAST), Pune, India, 19–21 December 2016; pp. 255–260.
15. Soualhia, M.; Khomh, F.; Tahar, S. A Dynamic and Failure-aware Task Scheduling Framework for Hadoop. *IEEE Trans. Cloud Comput.* **2018**. [CrossRef]
16. Trestian, R.; Shah, P.; Nguyen, H.X.; Vien, Q.-T.; Gemikonakli, O.; Barn, B. Towards connecting people, locations and real-world events in a cellular network. *Telemat. Inform.* **2017**, *34*, 244–271. [CrossRef]
17. Maheshwari, N.; Nanduri, R.; Varma, V. Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework. *Future Gener. Comput. Syst.* **2012**, *28*, 119–127. [CrossRef]
18. Shah, V.; Trivedi, H. A distributed dynamic and customized load balancing algorithm for virtual instances. In Proceedings of the 2015 5th Nirma University International Conference on Engineering (NUiCONE), Ahmedabad, India, 26–28 November 2015; pp. 1–6.
19. Gunarathne, T.; Zhang, B.; Wu, T.L.; Qiu, J. Scalable Parallel Computing on Clouds Using Twister4Azure Iterative MapReduce. *Future Gener. Comput. Syst.* **2013**, *29*, 1035–1048. [CrossRef]
20. Li, Z.; Yang, C.; Liu, K.; Hu, F.; Jin, B. Automatic Scaling Hadoop in the Cloud for Efficient Process of Big Geospatial Data. *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 173. [CrossRef]
21. Chen, Y.W.; Hung, S.H.; Tu, C.H.; Yeh, C.W. Virtual Hadoop: MapReduce over Docker Containers with an Auto-Scaling Mechanism for Heterogeneous Environments. In Proceedings of the International Conference on Research in Adaptive and Convergent Systems, Odense, Denmark, 11–14 October 2016; pp. 201–206.
22. Fu, Q.; Timkovich, N.; Riteau, P.; Keahey, K. A Step Towards Hadoop Dynamic Scaling. In Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications and IEEE 16th International Conference on Smart City and IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Exeter, UK, 28–30 June 2018; pp. 67–74.
23. Yadranjiaghdam, B.; Yasrobi, S.; Tabrizi, N. Developing a Real-Time Data Analytics Framework for Twitter Streaming Data. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 329–336.
24. Heine, G.P.; Woltron, T.; Wohrer, A. Towards a Scalable Data-Intensive Text Processing Architecture with Python and Cassandra. In Proceedings of the Seventh International Conference on Data Analytics, Athens, Greece, 18–22 November 2018; pp. 15–18.
25. Hwang, K.; Bai, X.; Shi, Y.; Li, M.; Chen, W.; Wu, Y. Cloud Performance Modeling with Benchmark Evaluation of Elastic Scaling Strategies. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 130–143. [CrossRef]
26. Trupthi, M.; Pabboju, S.; Narasimha, G. Sentiment Analysis on Twitter Using Streaming API. In Proceedings of the 2017 IEEE 7th International Advance Computing Conference (IACC), Hyderabad, India, 5–7 January 2017; pp. 915–919.
27. Sehgal, D.; Agarwal, A.K. Sentiment analysis of big data applications using Twitter Data with the help of Hadoop framework. In Proceedings of the 2016 International Conference System Modeling Advancement in Research Trends (SMART), Moradabad, India, 25–27 November 2016; pp. 251–255.