Cost-Aware Big Data Stream Processing in Cloud Environment

Ahmed Al-Mansoori(✉), Jemal Abawajy, and Morshed Chowdhury

School of Information Technology, Faculty of Science, Engineering and Built Environment, Deakin University, Geelong, Australia
{ajalma,jemal.abawajy,morshed.chowdhury}@deakin.edu.au

Abstract. The increasing size of big data and the speed with which it is generated have put a tremendous burden on cloud storage and communication systems. Network traffic and server capacity are crucial to having systems that are cost aware during big data stream processing in Software-Defined Network (SDN) enabled cloud environment. The common approach to address this problem has been through various optimization techniques. In this paper, we propose SDN based cost optimization approach to address the problem. Although SDN has been shown to improve cloud system performance, there is little attention given to SDN-based cost optimization approach to address the challenges of the increasing big data. To this end, we used Spark Streaming Processing approach (SSP). The proposed cost optimization approach is based on SDN within the cloud environment and focuses on optimizing the communication and computational costs. We performed extensive experiments to valid the approach and compared it with a Spark Streaming approach. The results of the experiment show that the proposed approach has better cost optimization than the baseline approach.

Keywords: Big data stream processing · Cloud computing · SSP · Cost-aware

1 Introduction

Big data is generated by a wide variety of resources such as Global Positioning System (GPS), Internet of Things (IoT) and social media systems [1]. With its potential to derive high-quality insights and improve data-driven decision-making, big data as a new paradigm has attracted increasing attention from researchers and practitioners. Appropriately exploited, big data will enable organisations to extract valuable insights, innovate growth and improve productivity. Therefore, Cloud computing has become the backbone for processing and analysing big data promptly. Cloud computing offers elastic on-demand resources such as computing, storage, networking and software [2,3]. This pool of resources is suitable for processing big data in real-time applications that demand real-time processing. With recent attention to big data stream processing (BDSP), how to handle data traffic and volume in cloud data centre has become a significant challenge. Especially the BDSP requirement for latency

L. Qi et al. (Eds.): CloudComp 2020, LNICST 363, pp. 120–136, 2021.
https://doi.org/10.1007/978-3-030-69992-5_10

and bandwidth issues brings new obstacles to cloud data centres. Moreover, the increasing size of big data and the speed with which it is generated has put a tremendous burden on cloud data centres. Besides, cloud data centres support numerous users, simultaneously causing a rapid increase in network traffic to the data centre, which also increases energy consumption [4]. Most importantly, the conventional network technologies and architectures that are most visible in the contemporary cloud data centres pose a severe challenge to the prospect of handling large volumes of data in real-time in cloud data centres [5]. Therefore, the need to meet traffic management for BDSP has raised interest for developing a mechanism that ensures the Quality of Service (QoS) to end-users such as delivering optimal services with as many minimal costs as possible [6]. A significant challenge is how to ensure the QoS for big data stream processing. A variety of approaches have been proposed to address these challenges. The primary practice is to push the data streams straight to a cloud data centre to optimise the processing the data when it reaches the data centre through approaches such as allocation of virtual machines [7] and optimisation techniques such as machine learning [8–10]. These approaches rarely consider the inherent BDSP issues that include a high level of latency, high level of data rates, and the infinite order of incoming streams [11]. Therefore, a robust framework that ensures QoS for big data stream processing is necessary [12].

A possible approach to address these challenges is to deploy a SDN. [13] shows that the data centres based on conventional network technologies and architectures are inferior to SDN-based cloud data centres because the later allows for the network to be managed dynamically and thus greatly enhance the efficiency of the network. As a result, interest in SDN-based cloud data centres has recently increased. SDN-enabled distributed open exchange framework for dynamic optimisation of end-to-end (E2E) and QoS paths selection has been discussed in [14].

[15] discuss a software-defined network (SDN) for big data stream processing. [12] propose an SDN-enabled network service for big data science suitable for campus or local area networks. [5] propose an SDN-based traffic management method for cloud data centre. Although an approach that utilises a software-defined network (SDN) for big data stream processing has started appearing [15], it is still at its infancy. Moreover, the challenges of cost optimisation while at the same time, ensuring that performance levels are highly maintained remains. To address these challenges, we propose an SDN-based framework that provides QoS in terms of cost optimisation for big data stream processing. We make the following contributions:

- A mathematical formulation of the cost optimisation while ensuring efficient performance.
- An SDN-based framework that ensures QoS in terms of cost optimisation for big data stream processing Results of the experimental evaluation to validate the proposed approach and its performance.

The rest of the paper is organised as follows. Section 2 covers the review of related works while Sect. 3 covers the problem formation. In Sect. 4, the system model is discussed while Sect. 5 presents the proposed stream big data processing framework. Section 6 covers the validation of results found, and Sect. 7 covers the conclusion.

2 Related Work

Cloud computing and big data stream processing have emerged as important areas of research, and they continue to receive a lot of attention among researchers. There have been several attempts to investigate cost optimisation issue in the SDN enabled cloud environment. Several researchers have tried to come up with efficient scheduling methods aimed at ensuring cost optimisation. Below is a review of the researchers that have performed an investigation on the issue and the shortcomings of their studies that determine the improvements that we shall make.

[9] described an approach that could be used to minimise cost for big data processing in data centres distributed in various locations. The author characterised data processing using "two-dimensional Markov chain" and derived the time to completion closed-form. Using this, the author computed a joint optimisation as an MINLP problem. The model was further based on graph construction using the function $G_v = (V_v, E_v)$. In the function, V_v stands for the tasks while E_v represents the data links. The authors covered several costs, including communication, server, and operation costs. They also focused on determining the time taken to complete a given task by taking into account computation and data transmission to determine the average time it takes to complete. Also modelled their problem as a mixed "integer non-linear programming" and proposed a way their problem could be linearised. This gives an insight into the areas where cost optimisation can be looked at so that the optimisation goals can be met. The shortcoming of this approach is that it never takes into account the fact that the servers have calculation and capacity limit limitations.

[7] defined a task flow graph algorithm denoted by Gt = (Vt, Et) where Vt represented all the tasks while Et denoted that data flow links that existed among the tasks. The algorithm helped the authors to determine the way of minimising communication cost in data centres that are distributed widely. The modelling framework proposed by the authors described "representative intertrack relationship semantics" in BDSP. The authors also formulated the "communication cost minimisation problem" for BDSP and came up with mixed-integer linear programming (MILP) and proved that the problem was NP-hard. Using the MILP, the authors came up with their solution. The shortcoming of this work is that the authors never took into consideration how the placement of the VMs will affect their solution. They assumed that the solution would work well under all conditions. However, VM placement is a crucial issue to the MILP solution that they proposed.

[16] proposed an ORCP algorithm for cost optimisation in cloud computing. The authors note that before ORCP algorithm, OVMP algorithm was already being used. OVMP algorithm helped in solving the problem of provisioning resources for cloud consumers and placement of VMs. With the ORCP algorithm, the researchers hoped to improve the number of stages involved in provisioning and reduce costs. Even though the authors never define an actual formula, they argue that using the ORCP algorithm is a right approach for large sets of scenarios. ORCP algorithm has been shown to be capable of performing in many studies. The authors formulated the stochastic programming model and took into consideration the price and demand uncertainty. They consider the demand uncertainty by the consumers and the price uncertainties from the cloud service providers. The authors hold that limiting these uncertainties is crucial to cost optimisation. The authors solved the ORCP problem using "simple average approximation approach" and this enhanced their ability to arrive at an optimum solution. The work was not without its shortcomings, the greatest of which the complexity of the programming language used. It is not easy to follow this algorithm to the conclusion and make sense of the entire solution.

Chen et al. [17] defined a model that was aimed at minimising computation and communication costs during big data stream processing. In the attempt to find the "optimal solution", the authors started with structuring an "ESWG" using the task "semantics of streaming workflows" and the diversity of prices of "geo-distributed data centres." The authors also formulated the "streaming workflow allocation problem" often known as MILP. Additionally, the authors pressed streaming workflow algorithms based on DC-ESWG and TC-ESWG and proved that their approach could work out better with lower costs. However, the shortcoming of the solution the authors provided was that it had a longer execution time with limited its efficiency. The algorithm has also not been tested in a real-world problem, and there is a lot of doubt on its workability. In other work Chen et al. [18] focused on cost-minimization in using transformation-based streaming workflow allocation where data centres are geographically distributed. They used streaming workflow allocation as the problem and addressed it with their proposed solution.

Zhao et al. [19] proposed a cost-aware scheduling algorithm on the cloud environment. The algorithm determined cost based on the total processing time. The algorithm determining time-based on the cost of resources and the number of processing tasks. The authors argued that to reduce the time; there is a need to ensure the cost remains low and the number of tasks processed at a time is also kept as minimum as possible. The function was: $T_{total} = max\Sigma_{t=1}^{n} T_{(r,t)}$. Where $T_{(r,t)}$ stands for the time cost of resources r to process task t and n is the number of tasks that have been assigned to r. The shortcoming of this research is that it never addresses the influence that the distribution of data has on schedule. Therefore, it would be difficult to ascertain the result gotten or even confirm that it is accurate or not.

In [20], the authors defined a big data processing architecture known as RedEdge, which used the principle of "data reduction" on edge to facilitate

big data new the edge. The algorithm used mobile IoT-devices as the primary processing platforms. The authors established that the model could reduce "big data streams" by up to 92.8% while optimising memory and energy consumptions on mobile devices. The biggest shortcoming of this model is that it never takes security and privacy into account. Therefore, there is no way of guaranteeing the protection of data that is streamed. This is a significant flaw that needs to be addressed because security must be given a priority regardless of the model that is used.

The models described above aim at providing an effective way of managing big data. They aim at improving the computation and storage process as a way of optimising costs. The issue that should be addressed in data processing is ensuring adequate and reliable data placement. SDN is a concept that is being embraced in the cloud environment. Without SDN, managing and provisioning networks was always a labour intensive and challenging process. SDN eliminates this pain by enabling administrators to use virtual resources in their network administration. Even though the several algorithmic developments have come with their benefits, their failures have also been well documented. Most of the algorithms that have been developed face issues of scalability after they have left the laboratory or simulation environment. Most of the algorithms developed are not fault-tolerant, and they lack the stability of a system that is ready to be deployed in the production environment. Additionally, those of the models that are developed assume that they will be executed offline. Some have also considered that their execution will be on a big batch of data. Most importantly, most of the algorithms to not provide for knowledge exchange to ensure that they can provide optimum solutions. The models designed have not been able to make use of the power of SDN. However, coming up with a model that is based on SDN supported cloud environment can help solve the shortcomings addressed above and increase scalability. The proposed model will aim at using the power of SDN to increase availability and scalability dynamically.

3 Problem Formulation

The problem being investigated in the research work is based on the establishment of a mechanism that ensures that BDSP in a cloud environment is enhanced using an elaborate and robust method of resource allocation. It is noteworthy that the existing methods of allocating resources in a BDSP environment need to have a balance between performance and the cost of resources incurred by the end-user. Majority of cloud implementations utilises implementation approaches that focus on the strategic placement of virtual machines. Numerous literature materials focus on the best strategies for placing virtual resources in a BDSP environment to ensure that performance is enhanced while at the same time cost utilisation is optimized. It is noteworthy that the placement strategies of VM resources should be based on the understanding of the cloud environment as well as the traffic requirements. Fundamentally, there lacks a comprehensive framework aimed at enhancing cost optimisation in BDSP based on a cloud utilising SDNs for the processing of real-time data. There is a need to develop a

comprehensive cost optimisation model that will ensure that BDSP in a cloud environment that uses SDN is optimized for maximum performance.

There is a excessive concern in ensuring that "big data stream processing" in the cloud environment is cost-effective. Adding SDN to control the entire process comes with added benefits, and there is a need to ensure that cost-benefit is also guaranteed. The issue is ensuring the challenge of cost optimisation is met in the long run.

To explain the problem, let us assume that we have S servers each having different capacity. The servers for a set $S = \{1, 2, 3, 4, 5.....s\}$. All these servers work at the same time, holding data that users may need during streaming. It is possible that these servers will have different computation capacities and powers, and they will, in turn, have different cost variations. Therefore, we will model the cost of streaming data. Let $C^i = \{1, 2, 3, 4, 5.....c^i\}$ processing costs, where C^i is the streaming cost for a given quantity of streamed data Q. We consider the resource consumption by CPU for every streaming job. This is necessary to establish the problem and determine how best it can be solved. We assume that every server has a capacity K and a processing speed P and takes time t to stream the data needed. A single server may need to handle multiple stream requests. Our primary interest is on cost optimisation. Therefore, we may want to know the time and cost of streaming a given quantity of information on each server and the resources used during this streaming. In simple terms, we want to know the cost of resources taken to stream data on a server concerning server capacity, the total number of streaming tasks, and the time taken to stream this data. We define the following equation to help model the cost issue:

$$C^i = Q_{si} K_{sn} P_{si} t_{si} \tag{1}$$

Therefore, we compute the different parameters for each server that leads to a reduction in the cost to be able to solve this problem. We want to reduce the total communication cost so that we can reduce the congestion on the network We will use the data gathered to draw up several cost components against each server. First, we will determine the server cost against the rate at which tasks arrive. We will also determine how communication costs compare with the number of SDN enabled servers that are being used. We will further examine the communication cost with the task arrival rate. Another cost component that will be determined is the operation cost, and it will also be compared with the number of servers and the task arrival rate. All these will be plotted in a graph to give a better comparative analysis that can be used to draw up meaningful conclusions. For us to be able to assign workflow streaming tasks in SDN supported cloud environment, we should consider several constraints. First, to total requests, data flow for task ti to be processed in the entire cloud environment will have to be optimum to ensure that the data flow is maximum. Therefore, we shall have:

$$\Sigma_{k=i}^{n} x(t_i, d) = E(t_p) \tag{2}$$

where $E(t_p)$ represents the incoming data flow for t_i.

Secondly, we need to consider the total computational resources that are required for the streaming workflows are ensure that they are kept at the best possible level. Finally, we need to consider the total of the network flows that are incoming and ensure that there is a good flow all around. The overall goal is to ensure that the total cost is kept as low as possible.

4 Proposed Model

In this work, the Spark Streaming Processing (SSP) model was used; the SSP is an executable object-oriented language that can help to model distributed systems. In addition, the flexibility offered by SSP is better than that of other ad hoc simulators as it has "parallel run-time support" and, it can support CPU memory and resources. In this work, the experiments were performed using CloudSim SDN based prototype application. The observations were made on SDN-enabled big data stream processing in the cloud for resource optimisation. The results are presented in terms of the number of physical servers, processing time considered at three levels such as local, distributed and random. The number of service requests and the delay caused are also observed. The proposed SSP model has an ABS data type with an associated size and an identifier. In the definition of the data types, the manes of parameters used in the constructor become "accessor functions." In the proposed model, an EmptyBatch function helps in determining whether the batch is zero or not, as shown below:

- type BatchID=int;
- data Batch =Batch (BatchID bID, int bSize);
- def Bool isEmptyBatch (Batch batch)=(bSize(batch)==0

This work focuses on batch modelling so as to keep the modelling simple. In this regard, it is assumed that the batch interval specified by users is equal to the block interval. The focus is also on modelling allocations that can be used to stream data and different job levels. The stage level processing is achieved through Stage and STJob data types. "STJob" is used to define a job that is not empty, while "Stage" defines those constraints that care used for execution at every stage.

- type StageId=String;
- data STJob=STJob (List<StageID> stages);
- data Stage=Stage (StageID stID, List<StageID> constr);

The Fig. 1 shows the architecture of the proposed SSP model.

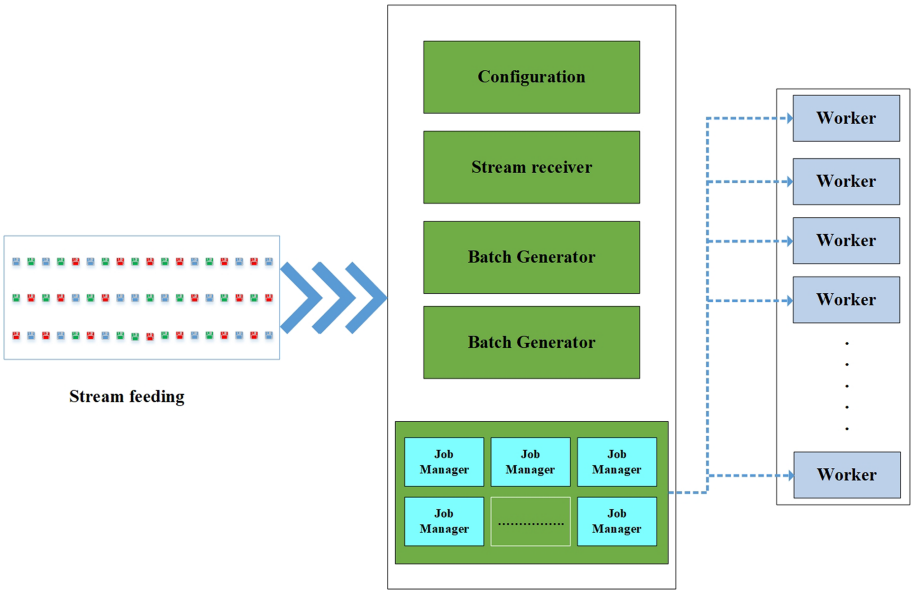

Fig. 1. SSP architecture

The architecture above has the main block on which users can configure their streaming applications. Figure 1 also has an underlying framework for Spark stream processing together with a "class SparkDriver" to help in modelling the Spark driver of the applications that are streamed. There is also a "class Worker" that can be used to model the work notes for applications that are used to process streams. The SSP has the following configuration parameters:

- The "job workflow specification" of the applications used for streaming.
- The cost of execution at every stage of data.
- The total number of worker nodes that are used in running applications.
- The resources that each worker node needs.
- The pattern for the data intervals
- The interval for the batches
- The maximum job numbers allowed to run concurrently.

The interface of the Spark driver has the following methods: jobScheduler, batchGenerator, streamReceiver, confSetup, and jobManager. When the SSP is launched, the confSetup method is invoked so that the number of worker nodes that have been requested can be created. It is the responsibility of the class worker to model every worker node. The class worker also has a method that is called exe. Using the modelling of the batch level, exe is executed by every stage of the streaming job. The cost expression that is used is e. The execution time of each step is determined by the expression "e" which denotes cost, and the speed of the CPU used for processing at the worker node. When the creation

of the worker nodes is completed, the confSetup method will trigger methods such as batchGenerator, streamReceiver, and jobSchedular so that the streaming application can start running. The data that has been streamed is then sent to the streamReceiver method that helps to hold it in the memory of the spark. Using the method batchGenerator, batches are generated according to prior pre-configurations for every batch interval. These batches are then inserted in a queue. Afterwards, joManager method is used to manage the batches that are processed.

The batchGenerator method helps to ensure that there is a constant flow of batch generation at the predetermined intervals. The technique uses a wait duration statement to suspend its operations, which resume after some time, depending on the set interval. The batchGenerator collects every data that has been received on the memory of the buffer and inserts it into a queue so that the buffer can be free to continue the processing cycle. Another method that is continuous is the jobScheduler method. It ensures that jobs are executed in an organized manner using the first-in-first-out method.

To help the model the function well, a computationally efficient heuristic algorithm is proposed. The aim here is to minimize costs and ensure that the execution time improves drastically. Thus, the algorithm is able to stream the cloud servers and the workflows. Firstly, two nodes that are supposed to help in the streaming of data are defined. The first node is the "stream data source" s node and the second node is the "stream data sink node" t. The locations of these nodes are usually predetermined. A virtual node V_v can be created for task T_t. Subsequently, a link is created between these virtual nodes using the structures of the patterns created by the streaming workflows; Dc denotes the links. The associated costs between server i and j is determined so that the communication costs between D_{ci} and D_{cj}, and the computational costs between D_{ci} and D_{cj} can be calculated. The main focus here is the data nodes, rather than the task nodes.

The complexity of the proposed model p, and its ability to work well depends on the number of workflows, the number of tasks within the workflows and the number of servers within an SDN-enabled cloud environment. Therefore, another algorithm that takes the number of workflows W, together with the latency requirement is viable. Thus, an approach that can be used in combining the server's nodes and the tasks appropriately. If that can be achieved, then the computational time can be reduced effectively. Therefore, streaming workflows will have to be partitioned to reduce their size. To achieve this, the workflow is first partitioned. Firstly, the "streaming workflow" F is mapped into its adjacent matrix $A(F)$ with n vertices d_n. Where n refers to the total number of tasks in F and can be denoted as

$$An(F) = a_{ij} n_{xn} \tag{3}$$

Where a_{ij} is used to refer to the value of communication requirement between two tasks i and j.

Next, the adjacent matrix $A(F)$ was translated to $B(F)$ using a data-clustering algorithm. Using matrix decomposition, it was possible to map $B(F)$

into submatrices denoted by $\{B-1(F), B-2_F), B_3(F).....B_k(F)\}$. The $B(F)$ matrix can then be used to divide F into several subgraphs that have smaller tasks. Finally, depending on the partitioning of the streaming workflow, we can have a definite graph for the algorithm.

5 Experiment

To evaluate the results of the proposed SSP model, a number of approaches were compared. The first approach was based on allocating workflows into SDN enabled cloud environment, and the next approach involved investigating the communication cost minimisation issue by comparing ow balancing and placement of virtual machines within the cloud environment. The focus is on communication and computational costs. Therefore, the performance metrics here are computational cost, which takes into consideration how much cost is incurred by a server in SDN-enabled cloud environment in executing streaming tasks. The cost was calculated as a total of all the tasks within the streaming flows. The next metric was the communication cost that determined how much traffic costs were incurred during the big data stream processing process within the cloud environment, especially following the SSP model. With these, the total costs that were incurred during the streaming process were determined. Here, the execution time for tasks during the streaming process was also taken into consideration.

The SSP model used over 300 streaming workflows which were made of different patterns. Every workflow had 50 tasks. The computation and communication costs were evenly distributed among these tasks, and they ranged from 0 to 50. The cloud environment chosen had close to 20 servers which each server is having a communication and computation capacity ranging from 10,000 to 100,000. There was a uniform distribution of communication and communication costs between the servers ranging from 0 to 0.2. It is essential to remember that the costs can vary depending on the server. For those streaming on the same network, the costs were set at 0.02, while those streaming from different networks was set at 0.1–0.2. The distribution was unequal.

In this work, the concept of spark streaming was employed, and a relatively stable computer with 8 GB RAM and 2.4 GHz processor speed was used. The computer was Intel (R) Core i7. The performance was evaluated using useful state approaches in terms of their abilities to achieve the required accuracy in terms of communication and computational costs. Tremendous efforts were made to ensure that close to 300 streaming workflows were averaged, while several parameters were varied including the type of the workflow, the requirement for data the capacity of the server, and the scale of the request of the workflow and the server.

6 Result and Dissection

First of all, the performance of the proposed model was evaluated by looking at the communication costs, computational costs and overall costs by averaging

the 300 workflows. Also, the costs were compared with the spark streaming, and the Fig. 2 shows that the proposed approach worked best in terms of saving on costs.

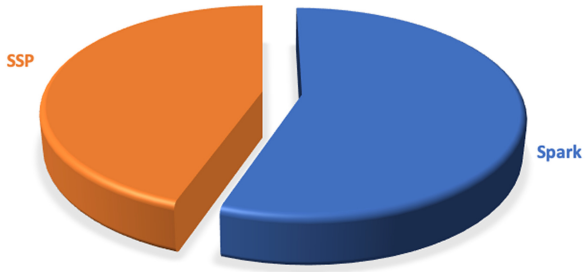

Fig. 2. Total cost

The result shows that the algorithm which was used helped in saving a significant portion of the total costs that were incurred in the entire process. Furthermore, the approaches were compared based on computational costs and communication costs independently, as shown in Figs. 3 and 4.

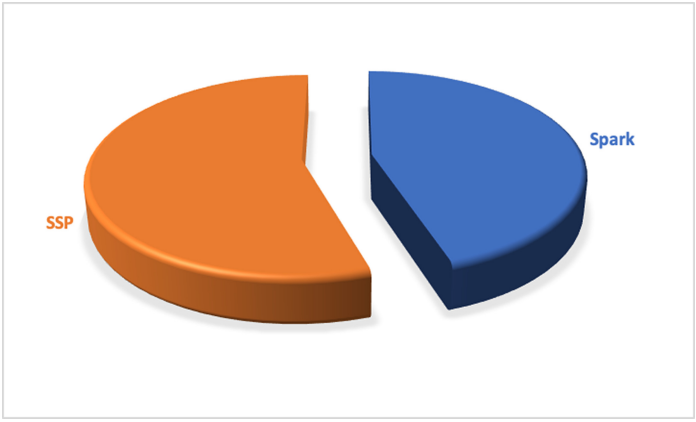

Fig. 3. Computation cost

Figure 3 show that the proposed approach performed well when only considering the communication costs, but the performance was poor in terms of

computational costs. This indicates that the method which was used can lead to a lot of cost savings, if only the communication cost is taken into consideration.

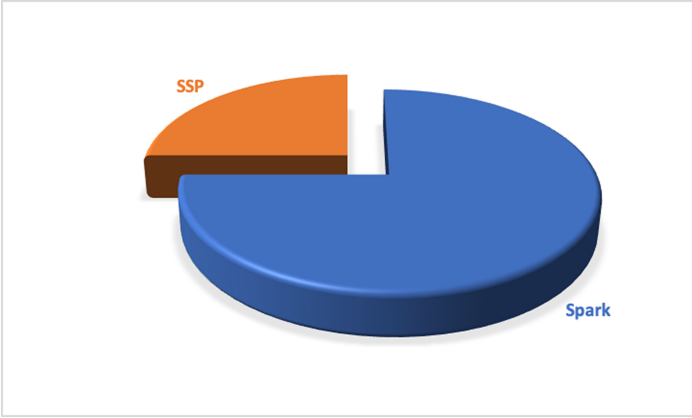

Fig. 4. Communication cost

The performances were also evaluated in terms of the time taken for execution. Figures 5 shows the batch processing time for both Spark Streaming and SSP.

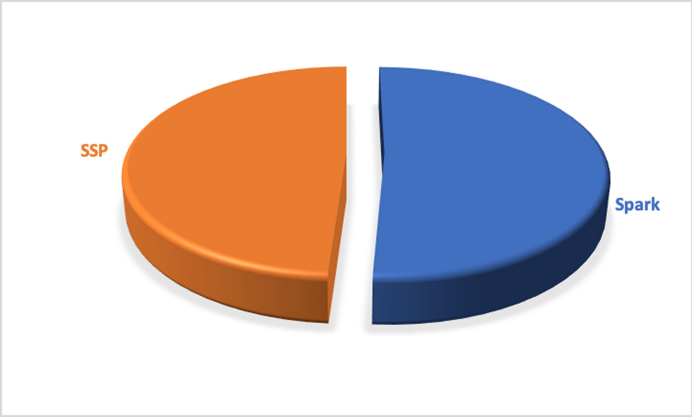

Fig. 5. Execution time

From the Fig. 5, it is evident that the proposed approach performed well in terms of the execution time, because it lowered the average batch execution time of Spark Streaming model. The performed better as it saved about 2,000 execution time compared with the spark streaming. Also, this approach reduced the

computational space for latency requirements during stream processing. Therefore, the results reveal that, the proposed approach performs better by lowering the computational costs and processing time.

Considering the workflow type: data streaming can be performed for different types of data and in different quantities but using a small quantity of computational power. It is with the used computational power that the communication cost incurred can be determined. Other streaming tasks can involve a small amount of data but still require a lot of power to be used in the computation. The value was varied in order to see the effect that it had on the communication and computational cost. The value was changed from 0.1 to 3. The results in the Fig. 6 show that our SSP model performed better at different values for all the values compared with the spark streaming. Also, the total cost was determined, and it was established that the SSP model performs better for all values, as shown below:

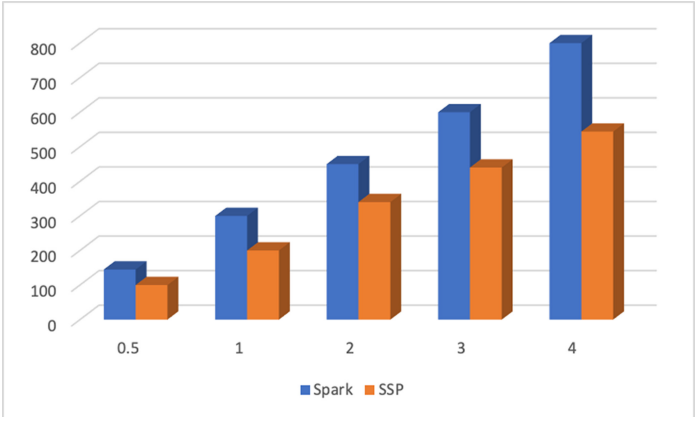

Fig. 6. Performance considering different workflow

As shown in Fig. 6, the value of a was varied at 0–1, and the SSP model performed better. This was further varied to 0.5, 1, 2, and 3, and for all those values, the cost performance was still significantly good. In fact, the performance of the proposed model continued to get better over the spark streaming model as the value of an increased.

Data Flow Requirement Impact: first of all, the performance of the proposed model was evaluated in terms of the two costs by varying the dataflow requirement between 5 and 25. It was observed that with an increase in the flow requirement, increase occurred in the processing time as more input data sets had to be processed. Therefore, the expectation was that the computational and communication costs had to increase. As shown in the Fig. 7, the total costs for both models increased as the data flow requirement increased. However, the

proposed SSP model achieved optimum performance with low data flow requirements. As the data flow requirements increased, the performance of the proposed model became almost as worse as that of spark streaming.

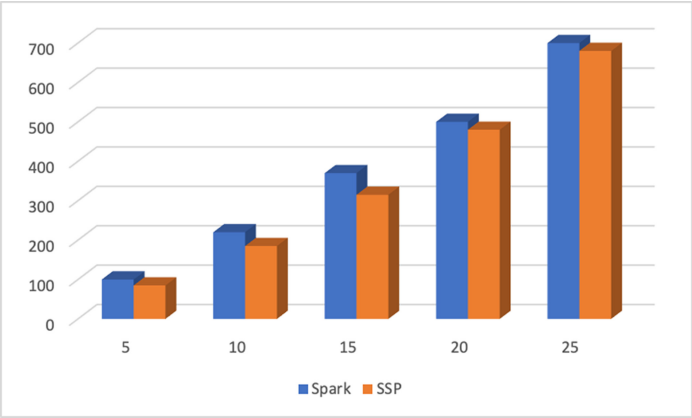

Fig. 7. Performance considering data flow requirement

Impact of the Requirement of VM: the requirements of the VMs were varied from 10 to 40 so as to note the effect which that would have on the communication and computational cost. As the requirements on VM_s increased, the costs were also expected to increase. The performance of the VM_s was almost the same for both the spark streaming and the SSP model. This could be attributed to the fact that the proposed model was not worked out to make the optimum use of VM_s. However, the fact that it was able to match the performance of spark streaming is considered as a big plus, and it showed that the VM_s could further help to optimize costs. Increasing the VM requirement towards 40 led to an increase in the total costs for both the communication and the computational costs. Also, the effect of increasing the VM requirement from 10 to 40 on the execution time was determined. It was noted that there was no significant change in the execution time, even as the VM requirement increased. In general, the lower the execution time, the better the performance.

Impact of the Sever Capacity: As with the previous steps, this step began by evaluating the effect that the capacity of the cloud servers had on the communication and computational costs, and afterwards, the impact on the execution time was investigated. In addition, the DC capacity was varied from 10,000 to 100,0000. The results of the experiment show that the lower the DC capacity, the higher the costs. This was true for both the proposed SSP model and spark streaming. However, the efficiency of the SSP model meant that it had better cost performances when compared with the spark streaming model.

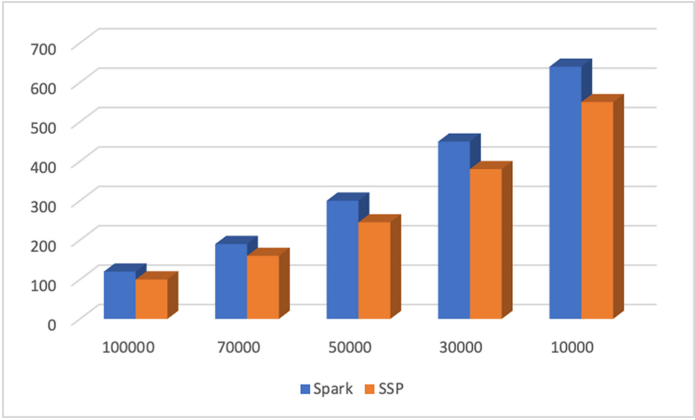

Fig. 8. Performance considering the Requirement of VM

As the Fig. 8, the performance of the proposed SSP model becomes better than the spark streaming model as the server capacity becomes lower. The gap in the total cost becomes more significant as the server capacity decreases. Therefore, it was concluded that the proposed model was more efficient at different capacity levels.

Further analysis which was performed focused on the impact on the scale of workflows and servers. All these, further revealed that the proposed model produced better performances overall, especially in terms of lower costs and better execution times. The evaluation focused on communication and optimisation costs. All the experiments performed revealed that the proposed model is better than the spark streaming model. It can be ascertained that the experiments were conducted with the utmost accuracy, and the results obtained were good. The model can therefore be taken as a viable solution to the issue of cost optimisation in the SDN-enabled cloud environment. In this regard, it has proven that this solution can indeed be taken into consideration and given some serious thoughts.

7 Summary

The SSP model was presented as an alternative to Spark Streaming with the aim of addressing the challenge of cost savings for big data stream processing in SDN-supported cloud environment. Also, the capabilities of the proposed model has been shown, and it has been established that the proposed model has the ability of help users adapt to the settings of their streaming applications, including stage execution cost and workflow. Apart from that, the model's ability to replicate the performance of Spark Streaming has been demonstrated, as well as its ability to provide users with a suitable alternative that enables them to save cost while maintaining the level of performance that they need. The experimental results showed that the proposed model could allow users to predict how their stream

processing will perform under various conditions. Big data stream processing is growing in importance, and with the concept of software-defined networking growing, a lot of changes can be expected. As technology changes, people will want cost-effective performance. Big data looks like a simple concept, yet it is complicated. There are also not enough professionals in the field with the ability to handle the technology effectively. A lot of resources are needed to collect, analyse and store big data entities. For this reason, cost-saving has become a significant concern.

There are several methods of resource optimisation, but one of the commonly used one is the reduction of the processing delay and ensuring that the streaming process is faster. For this reason, the proposed model will be more suitable and adequate for the cloud environment, because it involves ways through which batch processing delays can be reduced. The results further indicated that the FIFO approach could be more suitable for the optimization of resources on the cloud environment in processing data packets. Specifically, it covers the time used for packet processing. Users on cloud-based environments can use the model proposed here without doubt, because it has been tested thoroughly and compared against Spark Streaming which is one of the best models out there. Despite the numerous benefits offered by the proposed model, it is accompanied by some limitations, and as such, it is hoped that in the future, it can be fine-tuned in a manner that allows its operation on all the popular platforms. Currently, it can only work on a few of those platforms.

Additionally, that a model which is capable of addressing the security and privacy demands can be developed in the future. One of the most significant shortcomings of SDN technology is that it has not been able to address the security issue effectively. Due to this fact, many people have shown reluctance in accepting technology as a whole. The success of this model depends on the ability of the SDN to address its security shortcomings. It would be difficult to achieve cost optimisation without addressing security demands.

References

1. Abawajy, J.: Comprehensive analysis of big data variety landscape. Int. J. Parallel Emergent Distrib. Syst. **30**(1), 5–14 (2015)
2. Chowdhury, M., Abawajy, J., Kelarev, A., Jelinek, H.: A clustering-based multilayer distributed ensemble for neurological diagnostics in cloud services. IEEE Trans. Cloud Comput. **8**, 473–483 (2016)
3. Shojafar, M., Canali, C., Lancellotti, R., Abawajy, J.: Adaptive computing-plus-communication optimization framework for multimedia processing in cloud systems. IEEE Trans. Cloud Comput. **8**(4), 1162–1175 (2020). https://doi.org/10.1109/TCC.2016.2617367
4. Zhou, Z., et al.: Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms. Future Gener. Comput. Syst. **86**, 836–850 (2018)
5. Wang, Y., Wang, X., Li, H., Dong, Y., Liu, Q., Shi, X.: A multi-service differentiation traffic management strategy in SDN cloud data center. Comput. Netw. **171**, 107143 (2020)

6. Bouras, C., Ntarzanos, P., Papazois, A.: Cost modeling for SDN/NFV based mobile 5G networks. In: 2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pp. 56–61. IEEE (2016)
7. Gu, L., Zeng, D., Guo, S., Xiang, Y., Hu, J.: A general communication cost optimization framework for big data stream processing in geo-distributed data centers. IEEE Trans. Comput. **65**(1), 19–29 (2015)
8. Abawajy, J., Chowdhury, M., Kelarev, A.: Hybrid consensus pruning of ensemble classifiers for big data malware detection. IEEE Trans. Cloud Comput. **8**(2), 398–407 (2020). https://doi.org/10.1109/TCC.2015.2481378
9. Sowmya, T.S.R.: Cost minimization for big data processing in geo-distributed data centers. Asia-Pac. J. Convergent Res. Interchange **2**(4), 33–41 (2016)
10. Bhattacharya, M., Islam, R., Abawajy, J.: Evolutionary optimization: a big data perspective. J. Netw. Comput. Appl. **59**, 416–426 (2016)
11. Cao, H., Wachowicz, M.: The design of an IoT-GIS platform for performing automated analytical tasks. Comput. Environ. Urban Syst. **74**, 23–40 (2019)
12. Shah, S.A.R., et al.: AmoebaNet: an SDN-enabled network service for big data science. J. Netw. Comput. Appl. **119**, 70–82 (2018)
13. Adami, D., et al.: An SDN orchestrator for cloud data center: system design and experimental evaluation. Trans. Emerg. Telecommun. Technol. **28**(11), e3172 (2017)
14. Bagci, K.T., Tekalp, A.M.: SDN-enabled distributed open exchange: dynamic QoS-path optimization in multi-operator services. Comput. Netw. **162**, 106845 (2019)
15. Vicentini, C., Santin, A., Viegas, E., Abreu, V.: SDN-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming. J. Netw. Comput. Appl. **126**, 133–149 (2019)
16. Poobalan, A., Selvi, V.: Optimization of cost in cloud computing using OCRP algorithm. Int. J. Eng. Trends Technol. **4**(5), 2105–2107 (2013)
17. Chen, W., Paik, I., Li, Z.: Cost-aware streaming workflow allocation on geo-distributed data centers. IEEE Trans. Comput. **66**(2), 256–271 (2016)
18. Chen, W., Paik, I., Hung, P.C.: Transformation-based streaming workflow allocation on geo-distributed datacenters for streaming big data processing. IEEE Trans. Serv. Comput. **12**, 654–668 (2016)
19. Zhao, G.: Cost-aware scheduling algorithm based on PSO in cloud computing environment. Int. J. Grid Distrib. Comput. **7**(1), 33–42 (2014)
20. Habib ur Rehman, M., Jayaraman, P.P., Malik, S.U.R., Khan, A.U.R., Medhat Gaber, M.: Rededge: a novel architecture for big data processing in mobile edge computing environments. J. Sensor Actuator Netw. **6**(3), 17 (2017)