IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 16, 2018, accepted July 6, 2018, date of publication July 17, 2018, date of current version August 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2856623

A Multi-Dimensional Trust Model for Processing Big Data Over Competing Clouds

HADEEL T. EL KASSABI[1,2], MOHAMED ADEL SERHANI[2], RACHIDA DSSOULI[1], AND BOUALEM BENATALLAH[3], (Member, IEEE)
[1]Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC H4B 1R6, Canada
[2]College of Information Technology, United Arab Emirates University, Al Ain 15551, United Arab Emirates
[3]School of Computer Science and Engineering, University of New South Wales, Kensington, NSW 2052, Australia

Corresponding author: Mohamed Adel Serhani (serhanim@uaeu.ac.ae)

This work was supported by UAEU/UPAR under Grant 31T064.

ABSTRACT Cloud computing has emerged as a powerful paradigm for delivering data-intensive services over the Internet. Cloud computing has enabled the implementation and success of big data, a recent phenomenon handling huge data being generated from different sources. Competing clouds have made it challenging to select a cloud provider that guarantees quality of cloud service (QoCS). Also, cloud providers' claims of guaranteeing QoCS are exaggerated for marketing purposes; hence, they cannot often be trusted. Therefore, a comprehensive trust model is necessary to evaluate the QoCS prior to making any selection decision. In this paper, we propose a multi-dimensional trust model for big data workflow processing over different clouds. It evaluates the trustworthiness of cloud providers based on: the most up-to-date cloud resource capabilities, the reputation evidence measured by neighboring users, and a recorded personal history of experiences with the cloud provider. The ultimate goal is to ensure an efficient selection of trustworthiness cloud provider who eventually will guarantee high QoCS and fulfills key big data workflow requirements. Various experiments were conducted to validate our proposed model. The results show that our model captures the different components of trust, ensures high QoCS, and effectively adapts to the dynamic nature of the cloud.

INDEX TERMS Big data, big data processing, cloud computing, cloud selection, trust model, quality of cloud services, service evaluation, community.

I. INTRODUCTION

The recent evolution of information technology and the major paradigm shift of computation from the age of colossal massive machines to the omnipresent digital era have made information technology an important aspect of daily human activities. Nanotechnology, quantum computing, cloud-based computing, mobile computing and the new area of computation known as the Internet-of-Things (IoT) have generated massive volumes of structured and unstructured data. To glean their valuable insights, Big Data requires processing, analysis and storage. Big Data is not only defined by size, it is also characterized by multi V's; volume, variety, velocity, veracity, validity, volatility and value [1]–[4]. These special characteristics of Big Data introduce several challenges, such as data collection and integration problems, due to the data being distributed across diverse geographical locations. Moreover, the management, processing and storage of Big Data also present significant challenges considering the enormous volume and heterogeneous nature of the datasets, and traditional processing platforms are unable to efficiently handle such massively heterogeneous data volumes.

Cloud computing has emerged as a promising and powerful paradigm for managing and delivering computation, applications and services over the Internet [5]. It offers a large pool of easily usable accessible, and scalable virtualized resources capable of supporting Big Data key processes including storage, processing, and analytics. Hashem *et al.* [5] identified the correlation between cloud computing and Big Data, which exhibits good performance in distributed system environments, i.e., with respect to computer power, storage, and network communications. Amazon, eBay, Google, Microsoft and other leading Internet companies provide scalable cloud computing infrastructures suitable for Big Data processing such as MapReduce, the Google File System, BigTable and Dynamo [6].

Selecting the best cloud provider among this competing pool of cloud providers to store and process Big data is a challenging process. It is difficult for service consumers to decide which cloud provider to use as he/she may lack knowledge about whether the available cloud resource capabilities can handle Big Data tasks while satisfying a set of QoCS (Quality of Cloud Service) requirements. In addition, published QoCSs might be inflated for marketing purposes; hence, they cannot always be trusted. Furthermore, current trust models lack the flexibility to accommodate fluctuating user QoCS requirements. They also ignore the dynamic nature of trust, particularly in cloud environments. A QoCS is dynamically altered due to several factors such as changing demand levels (the number of service requests changes continuously over time) and the cloud provider's resource limitations. Thus, a trust model should adapt to the dynamic nature of cloud service usage. For example, a cloud server might be fully loaded at a specific time of day and lightly loaded at another time of day, which might be due to a periodical rush hour (e.g., end of month transactions) or an unpredicted increase in the number of service requests. Moreover, because the advertised QoCS information is often untruthful, trust can be more accurately evaluated using previously recorded QoCSs. Therefore, a Big Data client/application should typically perform a trust evaluation of a cloud provider prior to decide on transferring their critical data to the provider's cloud for processing or storage. Consequently, automating the decision-making process of cloud provider selection with an eye towards Big Data processing requirements and user QoCS preferences is highly desirable. Likewise, a comprehensive trust model – one that does not rely on the potentially falsely advertised QoCSs of cloud providers nor on historical records that cannot deliver accurate trust scores due to dynamic changes in cloud resources – is required.

In this paper, we propose a multi-dimensional trust model to evaluate the services of cloud providers based on: 1) the client's QoCS requirements, 2) the provider's current resources availability, 3) the historical records of his/her previous communications with the cloud service providers and 4) the community members trust score evaluation based on their own historical records of previous communication with the cloud service providers. It also, supports a community-based reputation provision wherein a community management system enforces a set of engagement and participation rules. Such a rule includes for instance: providing false information by a member will result in banning a member from the community. Before we detailed the proposed trust model, we thoroughly surveyed the related work on trust assessment, reputation evaluation, and trust score calculation. We then, provide a comprehensive classification and comparison between these different identified research initiatives. The majority of current trust models do not consider users' QoCS preferences and their contribution in the trust score evaluation. They are also non-dynamic and lack of real-time adaptability, which make them unsuitable for Big Data and the cloud environment. Depending solely on reputation can be misleading if the users are untrustworthy or subjective, especially given that different users have diverse opinions about the services provided. The majority of research initiatives on the service trust ignore information about the dynamic resource status of the cloud providing the service. Moreover, the existing trust models do not necessarily base their trust score evaluation on the QoCS attributes that are related to Big Data special characteristics and they produce unsatisfactory results with respect to Big Data workflow requirements.

The proposed model considers both the QoCS of SaaS and IaaS. The QoCS of SaaS is evaluated using history records logged by community members who have already experienced the service, whereas the QoCS of IaaS is evaluated by measuring cloud resources, i.e., memory and processing power. It is designed to differentiate between cloud providers based on their capabilities to process varying processing loads. Big Data tasks are modelled as a workflow where nodes represent Big Data tasks and arrows represent the transitions between two tasks. The QoS of the workflow are formally described and evaluated in order to be used to select the appropriate cloud service that guarantee these QoS. Selection algorithms are developed to consider also the three dimensions stated before: relies on the provider's advertised QoCS, assessments from community members and on the user's past personal experience with the cloud provider. Theses algorithms were developed to be lightweight and do not add an extra burden on the user, community members and cloud providers.

In the next section, we survey existing trust models, trust score computation and evaluation approaches. We also proposed a comprehensive classification and comparison of trust models using some key criteria such as computation method and trust evaluation scheme.

II. RELATED WORK

A. TRUST MODEL APPROACHES

Trust model approaches are classified into four main categories in [7]: self-managed case-based, SLA-based [7]–[9], broker-based [10]–[12], and reputation-based approaches [13]. These approaches are all based on continuously monitoring the SLA for the purpose of maintaining trust in a dynamic cloud environment. Other classification initiatives have relied on the perception of either the user or provider, or both, to define a trust model. For example, in [14], a trust model is proposed based on evaluating the functional and non-functional (QoCS) properties of cloud services from the perspective of both the provider and consumer. Noor *et al.* [15] classified the trust models into policy, reputation, recommendation, and prediction. Prediction models are convenient in case there is no previous historical interaction with the cloud service provider which haven't been recorded.

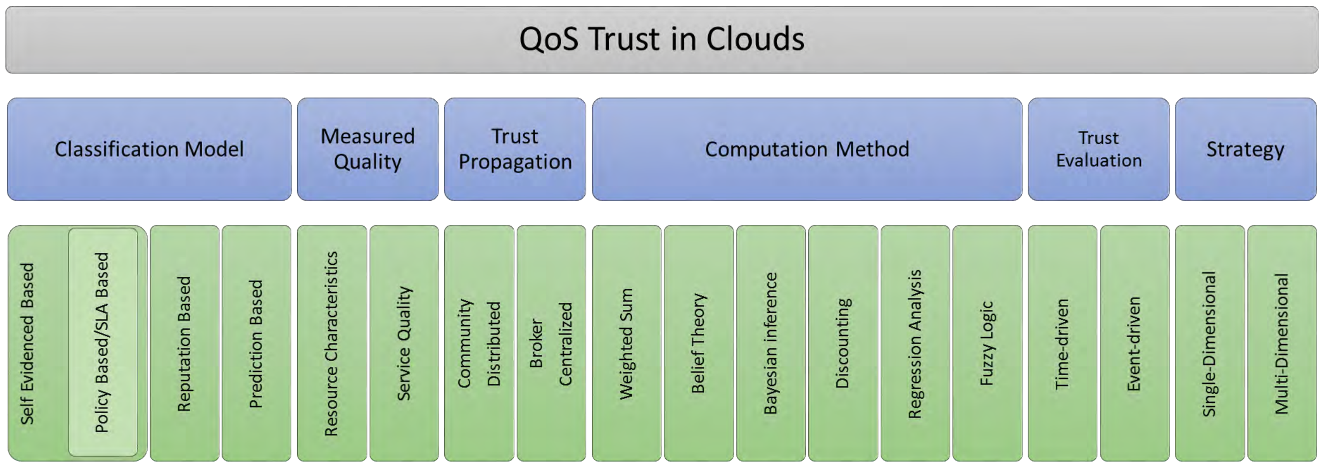


FIGURE 1. Classification of QoS Trust in Clouds.

In this paper, we propose a new classification of the trust model that integrates a self-based and reputation-based approach. A self-based trust model consults a recorded history of service provision to utilize the user's demonstrated experience with the service provider. However, a reputation-based trust model is based on the opinions and/or experiences of other users with respect to service providers. Reputation-based models can be further classified into service quality-based and resource quality-based models. A reputation service quality-based model evaluates the trust of the cloud Quality of Service and is typically performed on the SaaS layer. In contrast, a reputation resource quality-based model uses the quality of the cloud resources to evaluate the trust of the cloud service and is typically performed on the IaaS layer. Further QoCS trust models classification is illustrated in FIGURE 1 where we describe the QoCS trust in clouds in terms of trust based quality measured, trust propagation, computation methods, trust evaluation, strategy used. For quality-measured classification, the trust score in trust models depends on measuring the resources quality or capabilities or on measuring the quality of the service offered. Moreover, the trust propagation, describes the design of the trust collection method, if it is using a broker or collecting information from multiple neighboring users. We also classify trust models based on the computation algorithm used to calculate the trust. Another classification is based on the trust score evaluation trigger; which is either time-driven or event-driven. In addition, we classify the general strategy upon which the trust model is built on; is it single strategy, or adopting more than one strategy. More detailed description of these models is depicted in the following.

Self-based Trust Models: A trust approach proposed in [16] adopted historical usage records as the basis of trust evaluation and employed the Last-K algorithm wherein only the newest K records were used to calculate the trust score. However, this approach can result in decreased accuracy due to the limited number of attributes that are used, such as the time of invocation, while ignoring other important attributes, such as user input and user location. Gokulnath and Uthariaraj *et al.* [17] proposed an approach based on game theory to evaluate trust at boot load level combining both resources and user's perception. Another work used game theory to model trust for data-intensive cloud federations as depicted in [18]–[20].

Reputation-based Trust Models: We classify reputation trust models into service oriented and resource oriented according to the type of quality attributes used as basis to evaluate the trust score.

Service Oriented: Various research initiatives have focused on service quality-based reputation trust models. Muchahari and Sinha [21] recommended a registry and discovery system that keeps track of service providers and their feedback from credible service providers and users. The credibility of a service provider is measured by dividing the period of time over which the service is provided by the number of times the service is offered. However, user credibility is measured by the duration of their engagement with the service. A trust score is then calculated using the standard deviation, which is considered to be inversely proportional to trust. Kim *et al.* [22] evaluated the trust score of a cloud resource based on multiple QoCS attributes; however, the weights manually and nearly uniformly assigned. Hence, it was inflexible to user quality preferences for services. Hammadi and Hussain [23] adopted a fuzzy logic approach to calculate the trust score of a service provider based on user recommendations. The recommendation information was collected by users and stored at a third-party repository. The collected information was combined with SLA monitoring information, and the trust value and probability of service failure were calculated. Hammam and Senbel *et al.* [24] introduced a trust management system (TMC) for mobile ad-hoc clouds that calculated the reputation trust values of cloud nodes based on availability, neighbor evaluation, response quality and task completeness. In the approach proposed in [25], trust values were calculated based on QoCS attributes such as accountability, skills, service reliability, cost, performance, security, privacy and usability. Other researchers have introduced algorithms to calculate trust values based on QoCS attributes in accordance with users' experience with QoCSs, rather than their opinions about them [26]. The authors recommended two adaptive modeling algorithms, the rough set and induced ordered weighted averaging (IOWA) operator, to calculate trust scores. The advantage of the rough set is that, unlike traditional models, the weights of the QoCS attributes are not assigned subjectively. The advantage of the IOWA operator is that it uses time series for trust evaluation, thus adapting to the dynamic nature of the cloud. In the context of Big Data and cloud computing, Lin *et al.* [27] presented a category-based context-aware and recommendation incentive-based reputation mechanism (CCRM) for improving veracity and protecting data against internal attacks. A dynamic trust evaluation model with dual consideration of user preferences and false ratings is proposed in [28]. Manuel *et al.* [29] proposed a trust evaluation methodology for grid and cloud resources using a resource broker wherein a suitable grid or cloud is chosen according to user requirements. However, only simple factors, which did not cover the complexity of the trust evaluation, were used for their trust score evaluation [26]. A trust model was developed in [30] to enhance file transfers between the nodes of a private cloud. Their trust score was calculated based on node storage space, the operating system, network bandwidth and processing capacity.

Tang *et al.* [31] proposed a trust framework for cloud service selection named TRUSS. Their trust evaluation combines objective and subjective assessment based on QoCS monitoring and feedback ratings. Other proposals also combined objective and subjective models for evaluation of trust [32].

Resource Oriented: In [33], a trust model was introduced to improve the QoCS provided by the cloud, IaaS specifically, based on certain parameters such as the processing capabilities of the virtual machines (VMs), i.e., processing speed, fault rate, bandwidth and price. However, only IaaS was considered and no benchmarking was performed to compare the obtained trust results with other trust models.

Prediction-based Trust Models: Prediction-based trust models typically uses statistical techniques for trustworthiness evaluation and prediction [15]. They study the capabilities and the historical reputation of the service provider and predict how it will behave. These approaches use Fuzzy logic, Bayesian inference, or logistic regression models to estimate the trust of service providers as the probability of providing satisfactory QoCS to users [34]. These models are usually used when there is no previous historical interaction with the cloud service provider. They are also resilient to false reputation attacks especially the logistic regression models that are known to detect outlier values [35]. Bayesian inference is widely used as it considers trust as a probability distribution and is simple with strong statistical basis. However, the belief discounting technique is resilient to false attacks [34]. The fuzzy logic uses approximation for trust evaluation based on ranges between 0 and 1 rather than binary sets. It is widely used despite incurring some high implementation complexity and low malicious behavior detection [36].

B. TRUST SCORE COMPUTATION AND EVALUATION STRATEGIES

A variety of approaches were covered in the literature for reputation evaluation. A simple way to evaluate reputation scores is to calculate the difference between the number of positive ratings and the number of negative ratings. This easy-to-understand approach was used in eBay's reputation forum [37]; however, it can lead to ineffective results due to the simplicity of the method. A more sophisticated approach, used by many commercial websites such as Epinions and Amazon, calculates the average of all the ratings. A similar approach involves calculating a weighted average of all the ratings where the weights are based on the rater's credibility, age and distance between the new and existing ratings. Weighted sum trust calculation was also used in [32]. According to [34], other types of computational reputation models include Bayesian Systems [38], Regression Analysis [35], Belief Models [39], [40], Fuzzy Models [23], [41], [42] and Flow Models [43], [44]. However, not all of the aforementioned approaches are used for cloud provider trust evaluation because of unsuitability or simply untried.

The different computation methods are also associated with how the trust scores are scaled. The different scales for trust that are represented in literature include binary, discrete, nominal scale, and continuous values [45].

One problem with several trust score evaluation methods is that they are based on sophisticated and time-consuming mathematical models. This is unsuitable for a Big Data environment with its own special characteristics (multi-Vs). In addition, these time-consuming trust models, which are either service-oriented or resource-oriented, exhibit certain limitations. To the best of our knowledge, there is no comprehensive trust model that considers both service and resource quality.

Previous trust models are non-dynamic nature and lack of real-time adaptability, which make them unsuitable for Big Data and the cloud environment. Some base their trust only on reputation which can be misleading especially if the users are untrustworthy or subjective. Other trust models have used local trust and recommendation trust using weights that are not necessarily dynamic and suitable to the user's choice.

Trust score evaluation is related to the frequency of updating the trust score value. Studies in literature either undergo trust evaluation periodically to refresh the trust score, or it is done after a transaction, or even occurs upon request [34]. The periodic update is needed in cases of no existence of events or transactions leading to obsolescence of the QoCS information. A fade factor is used to determine how fresh the historical logs are. Some strategies give higher weight to newer records so that they reduce the emphases of the older records [16].

In this work, our trust model adopts the two selection strategies: periodic and event-driven. Periodic strategy relies on the cloud provider willingness to provide users with up-to-date information about the cloud resources. However, the event-driven strategy is executed upon receiving requests from users. The two strategies might be implemented concurrently to assure accuracy of data used to compute trust scores.

We classify the trust score strategy in this work as single-dimensional and multi-dimensional trust models. The single-dimensional trust models use only single strategy to evaluate the overall trust score such as considering service quality only or resources quality only. On the contrary, the multi-dimensional trust models combine more than one strategy to evaluate the trust, which is more comprehensive because it provides higher coverage of trust criteria. Examples of multi-dimensional trust models are [27], [28], and [32]. Lin *et al.* [27] proposed a category based and context aware reputation based trust model that integrates Vickrey-Clark-Groves recommendation incentive scheme for defending against internal attacks and bad mouthing reputation. The later does not provide an implementation and lacked a detailed system framework design. However, Li *et al.* [28] proposed a trust model for web service selection framework to allow user QoS preferences in addition to false ratings detection. Moreover, Nitti *et al.* [32] defined two models for trustworthiness management in the Social Internet of Things. Each node is responsible for calculating trust either subjectively or objectively but still no framework is represented in this work. Unlike the aforementioned proposals, our model use triple-strategy where cloud resources quality, self-experience and reputation strategies are used altogether to evaluate trust. In addition, the reputation assessments are also based on the user preference with emphasis on Big Data processing requirements.

In summary, most of the trust score evaluation in the related work described above, is not necessarily based on the Quality of Cloud Service (QoCS) attributes related to Big Data special characteristics. In addition, they are not dynamic in nature and lack the real-time adaptability. Furthermore, some of these models are based solely on reputation which leads to a weak trust assessment. However, our proposed selection model uses multidimensional strategies that dynamically captures Big Data characteristics and user quality preferences to assess the trust.

III. MOTIVATING SCENARIO

An epileptic patient need to be continuously monitored to detect seizures as soon as they occur to allow immediate intervention. Monitoring process should not restrain the mobility of patient both indoors or outdoors. Therefore, multi-channel wireless sensors are placed on the patient's scalp to record EEG signals and send these to a smart phone that allows a patient to move while being monitored. Since recorded data in continuous from different channels, it can result in a Big data (e.g. 128 EEG channels using a sensing frequency rate of 128 HZ generate around 1 GB of data every one-hour monitoring). Nevertheless, smart phones still lack full capabilities to handle Big Data. Hence, cloud computing technologies can efficiently enable acquiring, processing, analyzing, and visualization data generated form monitoring. FIGURE 2 describes the epilepsy monitoring workflow.

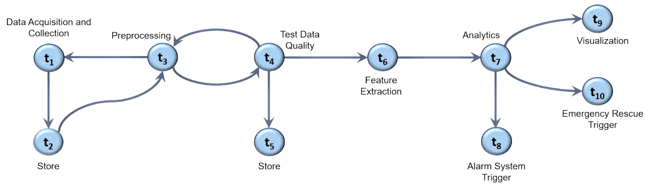


FIGURE 2. Epilepsy monitoring workflow.

Brain sensors collect the EEG signals, then sensory data are transferred to a smart phone or a back-end server where they are processed, analyzed, visualized in order to serve the seizure detection and/or prevention. The workflow is composed of ten main tasks as follows:

Task 1. Data acquisition and transmission, it is the process where the EEG data is acquired from the scalp by sensor electrodes that measures electrical activity of the brain and then transfer the signals to a computing environment for preprocessing or to a temporary storage.
Task 2. Raw data storage, it is the process of storing the raw EEG signals.
Task 3. Data preprocessing, it is the process of conducting some data cleansing and filtering activities to remove unwanted and noisy signals.
Task 4. Test data quality, it consists of conducting assessment activities of a set of data quality attributes including data accuracy, completeness, consistency.
Task 5. It is the process of storing the preprocessed data.
Task 6. Feature Extraction, it applies feature extraction and selection techniques to extract relevant features from the EEG signal to support the analytics.
Task 7. Data analysis where techniques are applied to the EEG data in order to extract meaningful information and insights that will support diagnosis and decision-making.
Task 8. In case a seizure is detected an alarm is triggered.
Task 9. Visualization task generates graphical reports to be viewed by different stakeholders
Task 10. Upon diagnoses of a seizure event, the emergency rescue task is triggered.

Big Data workflow aggregates different tasks that exhibit certain requirements such as optimized execution time, and efficient processing power. A workflow instance can be executed by a cloud service provided by one or more cloud providers [2]. Hence, the quality of a cloud workflow instance is a complex combinatorial problem. Accordingly, guaranteeing high quality workflow output from different quality dimension perspectives becomes very challenging. Quality models were designed in the literature to comprehensively support the lifecycle of cloud workflow instances. The main components of existing quality models in literature are time, cost, and reliability [1]. These QoS models use formal mathematical techniques to estimate overall QoS for a workflow process by determining QoS for each task and transition belonging/included to a given workflow. The following are some of the related QoS dimensions we used to evaluate the overall quality of the workflow:

Time: is the total time needed by a workflow instance to complete a Big Data job. Reducing the total execution time for a set of tasks is the ultimate objective of the user.

Cost: is the cost incurred when a workflow instance is executed. The cost is measured by the amount of money paid for executing that job.

Reliability: is the probability that the tasks will perform as per user expectation, it is measured based on success and failure rates.

We model a workflow as a directed acyclic graph DAG $w = (T, R)$, where T is the set of N tasks $\{t_1, t_2, \cdots, t_n\}$ and R is a set of M transitions between two tasks t_i and t_j so that t_j will not be executed unless t_i is completed. Tasks are represented using circles, and transitions are represented using arrows. We define P as a set of S possible paths in the workflow. Each path represents different sequence of tasks that are performed from the start to the end of the workflow instance and is represented by $P = \{p_1, p_2, \cdots, p_s\}$ where p_i is a sequence of tasks $t \in T$. Tasks t_i in a workflow Path p_i can follow a simple sequence, parallel sequence or/and contain loops. The task can be of two main types: processing task and storage task. Processing task is a task that performs a computational operation on the input data, while storage task is the task of storing the data. Every executed task $t \in T$, uses data with different sizes. In other words, a task can take different time depending on the data size processed by this task. Hence, each task is then represented in terms of task type and data size dsz: $t = (type, dsz)$.

The first quality dimension to model is time, the latter is measured from the start to the end of the workflow and it is the aggregation of the time of each task in each path P sequence. Hence, the time of the workflow can be measured as the maximum path time among the set of paths. The time taken to execute an atomic task is a function of data size dsz used by this task t_i and the average time taken by t_i to process one byte of data.

$$Time\,(t_i) = tAvg_i * dsz$$

$$Time\left(w_j\right) = \max_{0 \le j \le s}\left(\sum\nolimits_{i=1}^{N} Time(t_i)\right)$$

The second important quality dimension is the cost. The cost of the workflow is measured by adding the cost of each task executed along the workflow path. For data storage task, again the cost is a function of data size and time needed for storage. Hence, the cost is the amount of money paid to store the data. Conversely, the processing task is the amount of money paid for executing that task.

$$Cost\,(t_i)_{processing} = \sum_{i=1}^{n} cp_{t_i}, \forall t \in T_{processing}$$

$$Cost\,(t_i)_{storage} = \sum_{i=1}^{n} cs_{t_i} * dsz, \forall t \in T_{storage}$$

$$Cost(w) = Cost\,(t_i)_{processing} + Cost\,(t_i)_{storage}$$

In this paper, we use Amazon services' price as our reference: \$0.15 per Gigabyte per month for the storage resources. \$0.1 per CPU hour for the computation resources [48].

The third quality dimension is reliability, which is defined as the probability that the task can be completed successfully. Our proposed workflow model has been designed to cope with the situation were tasks can be handled by different cloud providers. However, in our trust model we only consider that tasks of the workflow will be handled by one cloud provider. The reliability of the workflow w is the product of the reliabilities of the cloud provider executing each task $t \in T$. It is the difference between the number of task requests and the failed tasks divided by the total tasks requests.

$$Reliability\,(w) = \prod_{i=1}^{n} R(t_i)$$

IV. MDTM ARCHITECTURE AND MAIN MODULES

The following sections describe the trust attributes and their evaluation in detail as well as the key modules of our proposed architecture.

A. TRUST ATTRIBUTES FOR QUALITY EVALUATION

Table 1 describes a mapping/relationship between Big Data properties and cloud quality metrics. These quality attributes are used to evaluate the degree of trustworthiness of the cloud providers.

Several trust attributes contribute to the trust score evaluation. To select a cloud provider to process Big Data, we should consider the ability of the cloud to process Big Data with respect to key Big Data characteristics such as volume, velocity and variety. Hence, it is essential to consider Big Data quality attributes that are essential factors in selecting a suitable cloud provider. We propose a mapping of some key Big Data characteristics to their related cloud quality attributes in Table 1. To incorporate the aforementioned QoCS attributes, we categorized them into four classes: low, medium, high and very high, with 1 being low and 4 being very high. Table 1 presents the following attributes:

Volume: The size of the data to be processed determines the class of this attribute.

Variety: This relates to the type of data to be processed. Class 1 comprises structured data, class 2 comprises unstructured data, and class 3 comprises structured and unstructured data types as a mix.

Velocity: This relates to the speed of the Big Data application. Class 1 indicates an offline data application

TABLE 1. Big data QoS attributes.

Big Data property	Metric (cloud)	Description
Volume	Data size trend	The size of the data to be processed determines the class of this attribute (1: low, 2:medium, 3:high, 4:very high)
	Disk space	Available disk space in the cloud
Velocity	Throughput	Total number of requests / (end time − start time)
	Response time	Actual execution time
	Availability	Number of received responses / number sent requests
	Resources (memory, processing power)	Available memory and processing power in the cloud
Veracity	Reliability	Task success ratio, the total number of task requests - the number of illegal connections and the number of denial of service incidents / total number of task requests
Variety	Data type	Categorized into four classes according to data type (1: structured, 2: unstructured, 3: mixed)

and Class 4 indicates the streaming of high speed data. Classes 2 and 3 represent intermediate speed levels.

The remaining QoCS attributes are measured according to common cloud characteristics used in the literature [25], [26], [49]. and behavior observed during interaction communications as follows:

Reliability: The task success ratio = the total number of task requests - the number of illegal connections and the number of denial of service incidents /total number of task requests.

Response time: The actual execution time = the time spent between sending a request and receiving the last byte of the response, in milliseconds.

Availability: The ratio of the number of received responses to the number of sent requests.

Throughput: The number of requests handled per second = (total number of requests / end time − start time) × 1000, where end time = last request response time and start time = first request start time.

Confidence: The degree of confidence in the response time is used to consider the delays caused by external factors on the client side. It is calculated as $1 - \sigma$ (response time)/μ (response time), where σ and μ are the standard deviation and mean of the response time, respectively.

B. USER SIDE MODULES

This section describes several modules that reside on the user side and are operated and managed by a user with the ultimate goal of determining which cloud service provider fits his/her QoCS preferences.

1) USER QoCS PREFERENCE MANAGEMENT

This module is responsible for managing the user's preferences in terms of the QoCS attributes and values that are required by the user and their acceptance levels. We developed a GUI to enable the user to input his/her specific requirements, which are collected and sent to the Trust module

2) TRUST MODULE

This is the core manager module responsible for collecting data from the User QoS Preference management module, analyzing the databases to evaluate the trust score for each cloud and providing a cloud selection decision to the user using the trust evaluation algorithm explained in Section 5. This module produces a trust value for each cloud service provider (CSP) and provides the user with the decision that yields the highest trust score. In order to reach this decision, the module runs the proposed algorithms on: 1) the Cloud_Spec database to generate a Cloud_Spec trust score for each cloud, 2) the Local History database to generate a direct reputation trust score for each CSP and 3) the IndirectReputation database to calculate an indirect reputation trust score for each CP. Subsequently, it applies the weights provided by the user to determine a final trust score for each CSP and finally selects the CSP with the highest trust score.

3) TRUST MONITORING MODULE

This module monitors communications with other clouds and collects the cloud's direct reputation information. A record is logged for each communication transaction exchanged between the user and the cloud provider. The log record contains QoCS information that can help to evaluate a cloud's trust score. This information is stored in a local history database called the Local History database. For each cloud, the log information includes multiple transaction logs, each of which contains the start time (invocation) of the transaction, data size, response time, cost and distance between the user and cloud and success status (success or fail).

4) LOCAL HISTORY DATABASE

This is the local history database containing the log information for each transaction invoked by the user to each cloud. It includes information about each cloud utilized by the user; the QoCS attribute values for each service, the time stamp of each task executed, information about the data being exchanged and the distance of the user to the cloud.

5) CSP REPUTATION MODULE

This module is responsible for collecting the cloud's reputation information from neighboring users, i.e., indirect reputation information. It sends an information request message to neighboring users in the community and handles the reply messages received. The request message contains the QoCS attributes to be evaluated and the preferred weight of each attribute. Each reply message contains a list of cloud providers and their corresponding trust scores calculated by the neighboring user according to the original QoCS user preference information parsed from the request message. This module also analyzes all the reply messages received and generates an average trust score for each cloud called the *avgIndirectScore* (more details will be explained in Section 5). This generated information is stored in the *IndirectReputation* database and is eventually communicated to the Trust module for the final trust evaluation.

6) CSP SPECIFICATION MODULE

This module is responsible for collecting the quality specifications and characteristics of each cloud. It sends message to all known CSPs requesting the specifications information including but not limited to the cost information such as the cost per second to use this resource, the cost to use memory of this resource, the cost to use storage of this resource, the cost per bandwidth, available memory, storage space and processing power. Then, this module analyzes the reply messages and stores all the parsed information into the *Cloud_Spec* database.

7) KNOWLEDGE BASED MODULE

This module is responsible for analyzing the data in Local History database and generating trust score for each CSP called directCPscore, which is then communicated to the Trust module for final trust evaluation.

C. CLOUD SERVICE PROVIDER MODULE

One module is needed on each cloud provider's side to handle resource information request messages. This module is called the ***User Request Handler***. It generates a reply message containing resource information such as available memory and CPU power. Reply message is received and handled on the user's side by the CSP Specification Module.

D. NEIGHBORS (COMMUNITY MEMBERS) MODULE

In the current study, we view reputation from a community perspective, which we will detail in this section.

1) COMMUNITY MANAGEMENT

Part of our trust evaluation depends on the cloud service provider's (CSP's) reputation within the community neighborhood. Because we are requesting reputation information from the user's neighbors, we must establish a degree of trust towards them. However, the neighbors require incentives to provide reputation information; thus, we propose a community management scheme to enforce this requirement. Community is defined in the Oxford dictionary as "the condition of sharing or having certain attitudes and interests in common". From this perspective, the user's community members have used mutual services or interacted with the same set of CSPs. In addition, they have a similar interest in obtaining CSP reputation information from other community members. Community management has been discussed in the literature in [49]–[51]. The community should be dynamic and adapt to the nature of the cloud environment. In order to protect the trust score against malicious reputation evaluations, we employ the following rules of engagement:

A third party maintains a database of community members' information. To join the community, a user sends a join request, which includes user authentication information, to the third party agent. Upon acceptance, a new community member is given an identification number and an initial reputation score. Community members have the following rights and responsibilities:

1) Provide honest information when requested to do so.
2) Other community members are responsible for providing CSP reputation information when requested to do so.
3) If malicious reputation information is provided by one of community members, a penalty is applied, i.e., reducing the community member's reputation score. This type of behavior is detected if the reputation score provided is considerably higher or lower than the average reputation score of the majority of the community members.
4) The reputation score is increased slightly each time a member provides CSP reputation information to other members.
5) It is difficult for a member to regain a favorable reputation score caused by a false accusation. A false accusation causes the reputation score to decrease dramatically, whereas any increase in the score is gradual in nature.
6) Members with low reputation scores do not receive reputation information from other members until they raise their reputation scores.

Each community member has a module to handle user reputation requests named ***User Request Handler***. This module receives request messages from the user, which contain the QoCS attributes and their weights. Upon receiving a request message, the module analyzes the information stored in the local copy of the ***Local History database*** and generates a trust score for each CSP, called the *indirectCPscore,* which is then populated in a reply message that is sent back to the requesting user, specifically, the ***CSP Reputation Module***. The ***Local History database*** consists of a local history log of communication with CPs similar to the user side history log.

V. TRUST MODEL

In this section we present our formal trust model for processing Big Data over a cloud platform. We formalize a Big Data service evaluation using the cloud's resource capabilities, its reputation among neighboring users and personal history of user experience. FIGURE 3 describes the proposed architecture.

Initially, we measure the resource capabilities of each cloud by collecting information from the cloud provider. We then collect the personal service history QoCS records

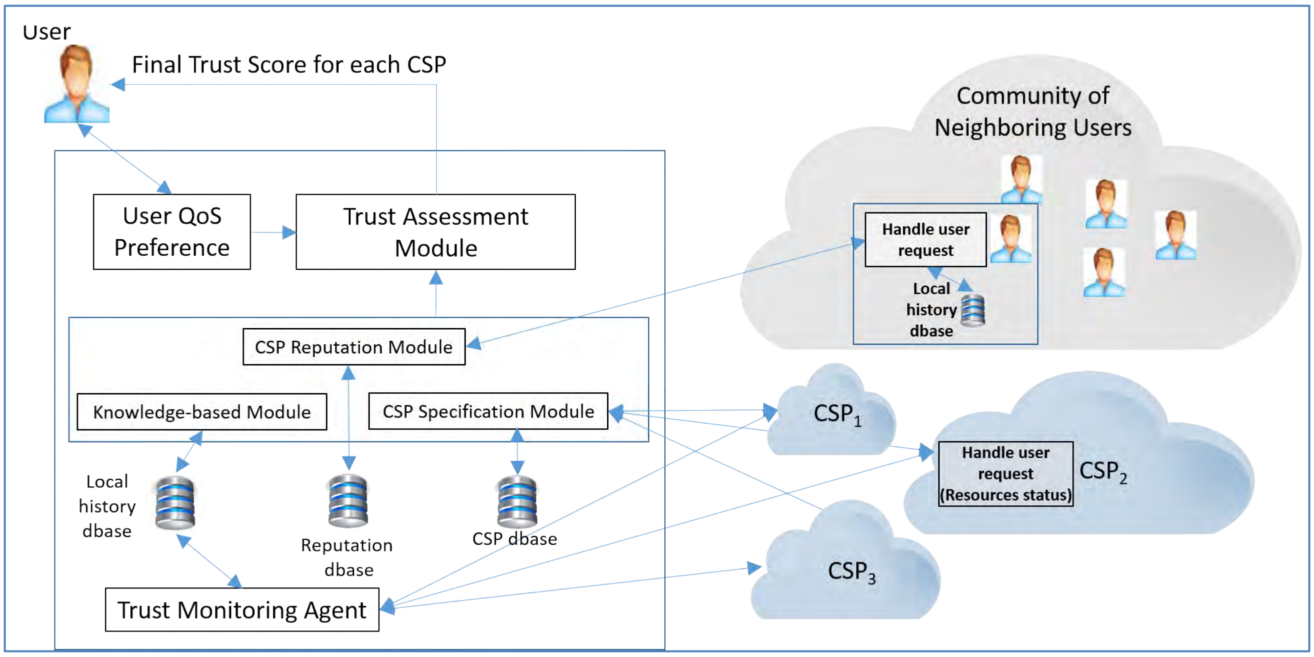


FIGURE 3. Multi-dimensional trust evaluation architecture.

followed by the provider's reputation from other users' historical QoCS records. The following sub-section details the three levels of our trust evaluation scheme.

A. CLOUD RESOURCE-BASED TRUST

The first stage of our trust evaluation scheme involves collecting the current resource characteristics from the potential cloud providers, i.e., memory, processing power, cost and the distance of the user from the provider. In addition, the user enters his/her personal preferences regarding the required Quality of Service. A weight is then assigned to each QoCS attribute according to the personal preference information. Next, a partial score is calculated for each cloud provider. We call this score the *CPcharacteristics* score. FIGURE 4 explains the algorithm for collecting cloud service provider characteristics. The attributes used to calculate the trust for these CSP characteristics include: Memory size, Processing power, Cost, Response time, and Data center parameters (price, processing speed, failure rate and bandwidth).

B. LOCAL SERVICE HISTORY-BASED TRUST

The second stage of our trust evaluation scheme involves calculating the history-based trust score. First, the user saves his/her service history records with each cloud provider in a database. We then calculate the cloud service providers' trust scores for each Quality of Service attribute. This algorithm is shown in FIGURE 5. The two factors that can affect the history-based trust score are: 1) the number of times the user has interacted with the cloud provider, and 2) the timeliness of the service history records expressed in terms of how newly they were recorded. The user experience factor is represented

Algorithm 1 Trust score based on CSPs resource characteristics algorithm

```
Input: CSPList //List of CSPs,
attributesList //List of QoS attributes of Cloud resources,
weights //Weight of each QoS attribute,
Output: Trust Score updated for each CSP
 1: procedure EVALUATECPCHARACTERISTICS(CSPList, attributesList, weights)
 2:     for all $csp \in CSPList$ do
 3:         sendRequest(csp, attributeList)
 4:     end for
 5:     while true do
 6:         recvResponse(csp, attributeVal)
 7:         $score \leftarrow 0$
 8:         for $i \leftarrow 1, n$ do
 9:             $score \leftarrow score + attributeVal[i] * weights[i]$
10:         end for
11:         $update(csp) \leftarrow score$
12:     end while
13: end procedure=0
```

FIGURE 4. Resource quality-driven CSP trust score calculation algorithm.

Algorithm 2 Trust score based on self historical interaction with CSP algorithm

```
Input: CSPList //List of CSPs, CSPServiceLog //Service Log of all CSPs,
attributesList //List of QoS attributes, WA //Weight of each QoS attribute,
WT //Weight of each time slot
Output: CSPListScore //Trust Score for each CSP
 1: procedure EVALUATECPSERVICE(CSPList, attributesList, WA, WT)
 2:     for all $csp \in CSPList$ do
 3:         for $attLabel \leftarrow 1, nAttributes$ do //in attributeList
 4:             $attScore \leftarrow 0$
 5:             for all $timeSlot \in CSPServiceLog$ do
 6:                 $attScore \leftarrow attScore + attributeVal[timeSlot] * WT[timeSlot]$
 7:             end for
 8:             CSPListScore[csp][attlabel] += attScore $\times WA[attlabel]$
 9:         end for
10:     end for
11:     return CSPListScore
12: end procedure=0
```

FIGURE 5. Self-historical interaction based CSP trust scores algorithm.

as a weighted score, the experience weight value, for each CSP and is calculated as follows:

$$\mathrm{E_{i,j}} = 1 - \mathrm{e}^{(-0.5\times \mathrm{Ni,j})} \quad (1)$$

where $\mathrm{E_{i,j}}$ is the experience of user *i* with the cloud service provider CSP *j*, and $\mathrm{N_{i,j}}$ is the number of history records the user *i* has stored on cloud service provider *j*, which may be outdated. Hence, we incorporate a time factor to calibrate the final score of each cloud provider using the following equation:

$$\mathrm{TF_{i,j}} = 1 - \mathrm{e}^{(-0.5\times 1/\Delta \mathrm{ti,j})} \quad (2)$$

$\mathrm{TF_{i,j}}$ is called the time fade factor with respect to user *i* to $\mathrm{CSP_j}$, Δt is the difference between the current and last interaction time between user *i* and CSP *j*.

The details of how we calculate a score for each cloud provider from their history of previous direct interactions with the user is shown in FIGURE 5. This score aggregates the scores of each QoCS attribute.

We calculate a final score using the user's personal preference weights for each QoCS attribute. We also assign higher weights to recent records over older ones using a weighted moving average algorithm.

The CSP score is then calculated for each CP, followed by the final *Max CPscore*. We use the following equation to calculate the *directCPscore* for each $\mathrm{CP_i}$:

$$\mathrm{directCPscore_i} = \mathrm{E}_i * \mathrm{TF}_i * \mathrm{CPscore}_i \quad (3)$$

C. COMMUNITY-DRIVEN REPUTATION-BASED TRUST

The third stage of our trust evaluation scheme addresses reputation-based trust. User trust has been extensively studied by several researchers.

The user requests the CSP scores from all the neighbors in the community according to: 1) the QoCS attributes chosen by the user and 2) the service history of the neighbors. To encourage users to provide rating information, incentives should be provided to the neighbors [52], such as receiving CSP reputation scores from the community to aid in their own decision making.

The reputation request message issued by the user contains a list of selected QoCS attributes and their user assigned weights. Upon receiving a request from a user, the neighboring users perform the following steps as detailed in FIGURE 6:

1) Calculate the experience weight value for each CSP ($\mathrm{E_i}$)
2) Calculate the time fade weight value for each CSP ($\mathrm{TF_i}$)
3) Calculate the CSP score for each CSP using the simple weighted Average (SWA) ($\mathrm{CPscore_i}$) using the weights embedded in the request message.
4) Return a reputation reply message containing a list of final CSP scores to the local user who originated the reputation request message.

The local user receiving the reputation reply message parses it to extract the scores for each CP, as shown in FIGURE 6.

After all the replies have been received by the user, the average score of each CSP among the *n* users who replied can be calculated:

$$\mathrm{avgIndirectScore} = \left(\sum\nolimits_{i=1}^{n} \mathrm{CPscore}_i\right)/n \quad (4)$$

D. FINAL TRUST SCORE CALCULATION

After collecting and compiling all the trust scores described in the previous cycle, i.e., the CSP resource-based trust score, the local history-based trust score and the reputation-based trust score, a final trust score is calculated for each CSP, and the max score becomes the selected CSP. The final score is calculated using a SWA method with user-assigned weights [53]. The SWA is used to compute the average of a group of numbers with asymmetrical importance. Other alternative is to calculate the weights according to existing values of quality attributes. However, we use the SWA to allow the user to enforce his quality preferences. There are different The following equation depicts the calculation of the final trust score:

$$\begin{aligned}\mathrm{FinalScore} = {} & ds * directScore + is * avgindirectscore \\ & + cs * CPcharacteristic \quad (5)\end{aligned}$$

where *ds*, *is*, and *cs* are weights given by the user, and

$$ds + is + cs = 1 \quad (6)$$

E. OVERALL TRUST SCORE FORMALIZATION

We formulate the trust score for each cloud provider as a Multi-Attribute Decision Making (MADM) problem [54] wherein the model is expressed as follows:

Step1: Model construction/initialization

$$\mathrm{CP} = \{cp_i \,|\, i = 1, 2, 3, \ldots n\} \quad (7)$$

$$\mathrm{A} = \{a_j \,|\, j = 1, 2, 3, \ldots m\} \quad (8)$$

$$\mathrm{W} = \{w_1, w_2, w_3 \ldots w_m\} \quad (9)$$

$$\mathrm{X} = \begin{bmatrix} x_{11} & \cdots & x_{n1} \\ \vdots & \ddots & \vdots \\ x_{1m} & \cdots & x_{\mathrm{nm}} \end{bmatrix} \quad (10)$$

where $cp_1, cp_2 \ldots cp_\mathrm{n}$ are the possible *n* alternative cloud service providers available to the user, $a_1, a_2, \ldots, a_\mathrm{m}$ represent QoCS attributes (criteria) such as reliability, availability and throughput. w_j is the weight (significance) of the jth

Algorithm 3 Trust score based on self historical interaction with CSP algorithm performed by neighbor

```
procedure EVALUATECPSERVICEBYNEIGHBOR()
    while true do
        recvRequest(src, cspList, attrList, WA, WT)
        cspListScore ← EvaluateCPService(cspList, attrList, WA, WT)
        sendReply(src, cspListScore)
    end while
end procedure
```

FIGURE 6. Indirect trust score (reputation) algorithm performed by neighbors.

attribute and x_{ij} is the performance rating of the i^{th} alternative (cp) with respect to the j^{th} attribute, which is calculated using a model explained later in this section. In this model, a higher score is assigned to the cloud provider with the highest performance rating, which preferably maximizes the j^{th} attribute.

Step 2: Construct the normalized decision matrix. This step is required to enable comparable attribute values with different scale units, and we normalize these values on a scale from 0 to 1. It is important to note that some attributes, such as throughput, have preferably high values, whereas others, such as cost, have preferably low values. Hence, to normalize these attribute values easily and fairly, it is useful to normalize them using Eq. 11 when a high value is preferred (beneficial attribute) and Eq.12 when a low value is preferred (non-beneficial attribute).

$$r_{ij} = x_{ij}/x_{ij}^{max} \text{ (row)} \tag{11}$$

$$\text{Or } r_{ij} = x_{ij}^{min}/x_{ij} \text{ (row)} \tag{12}$$

Step 3: Construct the weighted normalized decision matrix. We need to assign different weights for each attribute to allow preference towards quality attributes over other attributes. The weights are assigned based on user preferences and the nature of the Big Data workflow application. The values of the weighted normalized decision matrix are calculated using the following equation:

$$v_{ij} = w_j * r_{ij}, \quad \text{s.t.} \sum_{j=1}^{m} w_j = 1 \tag{13}$$

Step 4: Calculate the score of each alternative (cp).

$$score_i = \sum_{j=1}^{m} v_{ij}, \quad i = 1, 2, 3, .., n \tag{14}$$

Step 5: Select the best alternative (CP):

$$CPbestscore = \max_{0 \le i \le n} score_i \tag{15}$$

Here, the x_{ij} in the matrix (10) are the CSP partial scores for each attribute calculated from the history records. Following the same model used in an MADM problem, these partial scores are calculated following these steps:

Step 1: Model construction for calculating the x_{ij} in the above matrix (10):

$$A = \{a_j \,|\, j = 1, 2, 3, \ldots m\} \tag{16}$$

$$T = \{t_z \,|\, z = 1, 2, 3, \ldots nt\} \tag{17}$$

$$WT = \{wt_z \,|\, z = 1, 2, 3, \ldots nt\} \tag{18}$$

$$Y = \begin{bmatrix} y_{11} & \cdots & y_{m1} \\ \vdots & \ddots & \vdots \\ y_{1nt} & \cdots & y_{mnt} \end{bmatrix} \tag{19}$$

Where a_1, a_2 ...a_m, are the m selected QoCS attributes, $t_1, t_2, \ldots, t_{nt}$ are the different times at which the attributes were measured and nt is the number of time slots. We assume that $t_z < t_{z+1}$, for all $\{z = 1, 2, 3 \ldots nt\}$. Y_{jz} are the performance rating values of attribute j at time z.

Step 2: Construct the normalized decision matrix as explained previously. For simplicity, we only describe the beneficial attribute:

$$rx_{jz} = y_{jz}/y_{jz}^{max} \text{ (col)} \tag{20}$$

Step 3: Construct the weighted normalized decision matrix wherein higher weight is given to relatively recent attribute values and lower weight is given to older values. A higher value of z gives a higher weight.

$$vx_{jz} = wt_z * rx_{jz}, \quad \sum_{z=1}^{nt} wt_z = 1 \tag{21}$$

Step 4: Calculate the score of each alternative (attribute).

$$x_j = \sum_{z=1}^{nt} vx_{jz}, \quad z = 1, 2, 3, \ldots, nt \tag{22}$$

VI. EXPERIMENTS AND EVALUATION

In this section, we present both a formal evaluation and simulation experiments as follows: 1) we conduct a formal evaluation of our algorithm's complexity, 2) we formally evaluate the communication overhead generated from the execution of our multi-dimensional reputation scheme and 3) we simulate the environment of cloud selection-based reputation models and conduct various experiments to validate our trust model.

A. THEORETICAL EVALUATION

1) ALGORITHM TIME COMPLEXITY EVALUATION

It is worthwhile to evaluate the time complexity of each of our proposed algorithms as it measures the algorithmic efficiency, which has an impact on execution time. To evaluate algorithm 1 of FIGURE 4 the execution time depends on the parameter N, the number of CSPs, and parameter K, the number of QoCS attributes. Therefore, the time complexity is on the order of O (N.K). Though, algorithm 2 of FIGURE 5 depends on the number of CSPs (N), the number of QoCS attributes (K) and the number of time slot records stored (L). Accordingly, the time complexity of this algorithm is on the order of O (N.K.L) because there is no relationship between N, L and K. Moreover, in algorithm 3 in FIGURE 6, the time complexity depends solely on the number of requests R received by neighboring users in the community; thus, the time complexity is on the order of O (R). Again, for algorithm 4 in FIGURE 7, the time complexity depends on the number of neighbors who replied to the requests, the maximum number of community members (M) and the number of CSPs (N). Hence, the time complexity is on the order of O (N.M). In overall, all proposed algorithms exhibit a high efficiency and low execution time, which will not affect the performance of our Multi-dimension trust-based Cloud selection approach.

2) COMMUNICATION OVERHEAD EVALUATION

We evaluate the communication overhead for each evaluation strategy as the number of messages exchanged for the purpose of trust evaluation.

Algorithm 4 Indirect trust score (reputation) calculation algorithm performed by user

```
Input: $nbrList$ //List of neighboring users $CSPList$ //List of CSPs, $attrList$
//List of QoS attributes, $WA$ //Weight of each QoS attribute,
$WT$ //Weight of each time slot
Output: $localCSPScoreList$
procedure EVALUATEINDIRECTTRUST($nbrList, cspList, attrList, WA, WT$)
    for all $nc \in nbrList$ do
        $sendRequest(nc, cspList, attrList, WA, WT)$
    end for
    while $true$ do //with timeout limit
        $recvResponse(nc, cspListScore)$ //each record (csp, score)
        // update the local CSPListScore with the neighbor score
        for all $csp \in cspListScore$ do
            localCSPScoreList[csp].totalScore += cspListScore[csp]
            localCSPScoreList[csp].nReplies += 1
            localCSPScoreList[csp].score = totalScore/nReplies
        end for
    end while
    return $localCSPScoreList$
end procedure=0
```

FIGURE 7. Indirect trust score (reputation) algorithm performed by user.

Our comprehensive trust model requires messages to be exchanged between both 1) the customer and cloud providers and 2) the customer and neighboring users in the community. Using this strategy, we have 2 partial trust score evaluations requiring message exchanges. For the cloud resource-based trust score evaluation, the client must send a QoCS information request message to each cloud provider and receive a response message that includes the requested information. For the reputation-based trust score evaluation, a request message is sent to the neighboring users, and a response message is sent back carrying the required trust score values for each CSP. The total number of messages can be calculated using the following formula:

$$total\ number\ Msgs = 2 * nCSPs + 2 * nNeighbors \quad (23)$$

Evaluating the size of each message is performed following the below calculations:

First, for the cloud resource-based trust evaluation, the communication is expressed by the amount of data being transmitted between the CSP Specification module and the set of cloud providers. We measure the total size of the exchanged messages using the following formulas:

$$sizeReqMsg = nQoSAttr * sizeQosAttrName \quad (24)$$

The size of an attribute name is one byte (assuming that the maximum number of QoCS attributes is 2 to the power of 8, or 265 attributes).

$$sizeRspMsg = nQoSAttr * (sizeQosAttrName + sizeQosAttrPerformanceVal) \quad (25)$$

The size of an attribute value is 4 bytes, which is a double number.

$$sizeAllMsgs = nCSPs * (sizeReqMsg + sizeRspMsg) \quad (26)$$

FIGURE 8 depicts the communication overhead generated in Kbytes per number of QoCS properties used in the trust evaluation. Second, for reputation-based trust evaluation, we evaluate the communication overhead by calculating the amount of data exchanged between the customer and neighboring users in the community, i.e., between the CSP reputation module hosted at the customer side and each CSP reputation module hosted at each neighbor. The following formulas express this strategy:

$$sizeReqMsg = nQoSAttr * (sizeQosAttrName + sizeAttrWeight \quad (27)$$

$$sizeRspMsg = nCSPs * trustScorePerCSP \quad (28)$$

$$sizeAllMsgs = nNeighbrs * (sizeReqMsg + sizeRspMsg) \quad (29)$$

FIGURE 8, shows that communication overhead is proportional to both the number of neighboring users in the community and the number of selected QoCS attributes. With 20 cloud providers, 100 active community members and 15 selected QoCS attributes, the calculated overall communication overhead was nearly negligible (15 Kbytes). This proves that our trust model is lightweight because it does not incur a heavy load in providing cloud providers trust scores prior to the selection of the best CSP and guarantees optimal adherence to QoCS user requirements; thus, does not affect the performance of the overall solution.

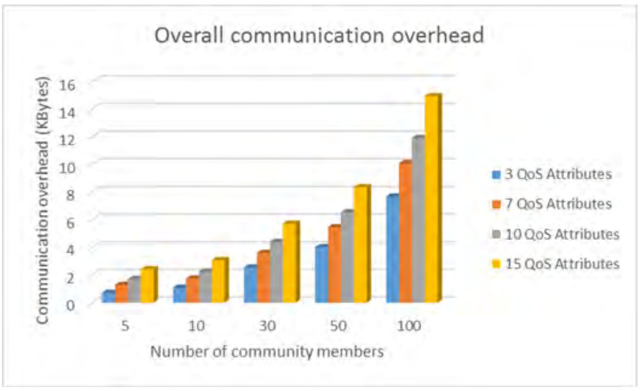


FIGURE 8. Overall communication overhead.

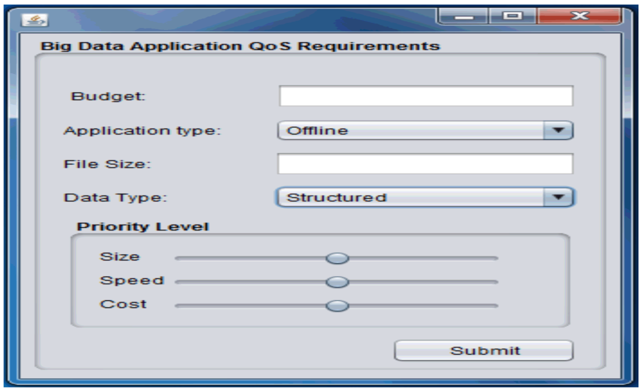
B. EXPERIMENTS

In this section, we describe the experimentations we have conducted to evaluate our proposed trust model architecture. We explain experimental setup, and then we describe the simulator system including all modules. Then we explain the scenarios of our experiments along with results interpretations. Finally, we provide a discussion of our results.

1) ENVIRONMENTAL SETUP

We considered the following default simulation parameters:

- Number of clouds: 1 to 50 clouds
- Number of nodes within each cloud: 1 to 100 nodes.
- QoCS specification file: budget, availability, Big Data application type, file size and priority level.


Big Data Application QoS Requirements

Budget:
Application type: Offline
File Size:
Data Type: Structured

Priority Level
Size
Speed
Cost

FIGURE 9. User interface for the collection of big data QoS requirements.

- Cloud properties: proximity, average node performance and unit storage price.
- Node properties: available resources, memory, disk space, processing power, round trip delay (RT) and bandwidth.
- QoCS attributes: data size, distance, cost, response time, availability and confidence.
- Number of community members: 3 to 100 neighbors.
- Reputation database: 20 timely history records for each CSP local to each community member.
- A computer desktop with Intel CoreTM i7-3770K CPU @ 3.40GHz with Turbo Boast, 32GB of DDR3 RAM, 1TB hard drive, and 64-bit operating system.

2) SIMULATOR SYSTEM DESCRIPTION AND MODULES

We developed a simulator in Java to test our proposed trust model. Our simulator implemented all four of the trust evaluation algorithms described in Section 4. We also developed a web-based application to help develop and design the workflow proposed earlier by collecting QoCS preferences from the client. It is also used for generating the weight values required for the overall trust score calculation shown in FIGURE 9.

The following evaluation is intended to prove the applicability of our proposed trust model for processing Big Data over competing clouds by: 1) evaluating whether the Big Data application QoS is significantly considered when selecting the appropriate cloud from the existing clouds, 2) showing the effect of each strategy that our comprehensive weighted trust score (CWTS) evaluation depends on while varying the weight of each strategy and monitoring the selection results, 3) evaluating whether our trust model is able to detect malicious trust ratings and react accordingly, 4) evaluating the effect of deleting one of the strategies that compose our trust evaluation model and 5) evaluating the scalability of our model towards supporting a large number of selection requests. Before we detail the evaluation scheme, we first describe our simulation implementation and component structure, our set of scenarios and the results that were obtained from executing the experiments. FIGURE 10 describes the main components of our simulation, including the client user interface, QoS Manager, cloud selection manager and cloud providers, as well as neighbor components (e.g., other users). In these experiments, we used CloudSim, which provides a generalized and extensible simulation framework to generate data to populate the cloud reputation databases, including direct and indirect reputation information. We also used the cloud characteristics data of CloudSim to populate the cloud spec database.

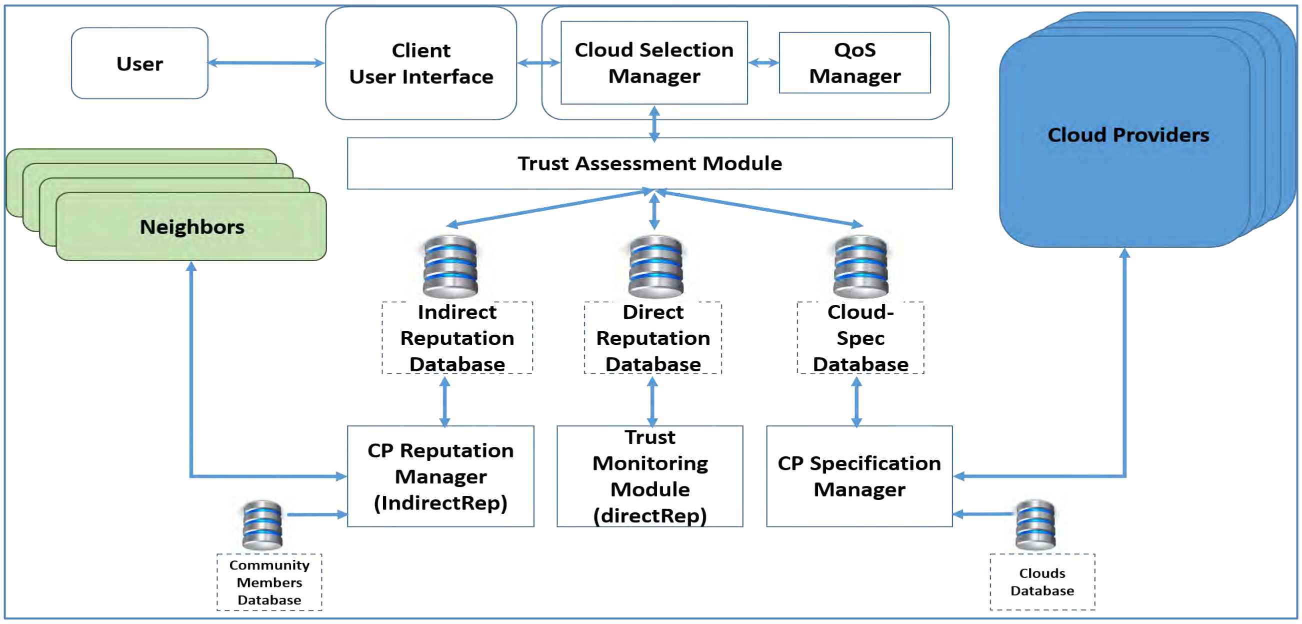


FIGURE 10. Simulation system components.

Each component can be described as follows:

- *CSP Reputation Manager:* Implemented according to the description in the architecture section. It is responsible for collecting the clouds' reputation information from neighboring users.
- *Clouds database:* The database that keeps track of the candidate cloud service providers.
- *Community Members database:* The database that keeps track of the neighbors; i.e. each community member.
- *Cloud Provider module:* This component simulates a cloud provider. It uses static log files collected by multiple runs of CloudSim, each with different cloud configurations and multiple users. We collect the log information to populate our databases. CloudSim provides a generalized and extensible simulation framework that enables modeling, simulation and experimentation with emerging cloud computing infrastructures and application services, allowing its users to focus on specific system design issues without handling the low-level details related to cloud-based infrastructures and services [55].
- *Neighbor Component:* Simulates a neighboring user that uses the log generated by CloudSim runs to populate the *Local History* database. Each neighbor object is responsible for responding to reputation requests from other neighboring users by analyzing its own *Local History* database.
- *Indirect Reputation database, Direct Reputation (Local History) database*and *Cloud Spec database:* Databases containing the QoCS information explained earlier in the architecture section.

3) SCENARIOS AND INTERPRETATION OF RESULTS

We built a simulator in Java and implemented the trust evaluation algorithm described in Sections 4 and 5. We developed a web application to collect QoCS preferences from the client and generate the QoCS specification file used by the developed algorithms.

We generated a series of client ad-hoc queries on Big Data sets that requires different QoCS properties, values and prominence. The client provides this QoCS information, including the weights of each QoCS attribute, via the application interface.

We evaluated whether the cloud selection manager module performed the appropriate selection of clouds that satisfy the client QoCS requirements. We also evaluated our comprehensive weighted trust score (CWTS) evaluation algorithm against three other single-strategy algorithms. The first algorithm generates a trust score based on physical cloud characteristics and qualifications in terms of capacity, memory and processing power (*Cloud_Spec*). The second algorithm generates a trust score based on the calculated direct reputation, which consists of the logs of the past interactions between the Big Data application client and the cloud provider (*Direct_Rep*).

The third algorithm generates a trust score based on calculated indirect reputation, which is the average trust score generated by the neighboring users in the community.

The neighbors collect reputation information during their communications with the cloud provider (*Indirect_Rep*). We first ran the CWTS algorithm giving equal weight to all three of the aforementioned strategies. We then ran the CWTS algorithm giving 100% weight values to the *Cloud_Spec, Direct_Rep* and *Indirect_Rep* strategies over the others, respectively. We also considered the FIFO strategy in some of our experiments. In the FIFO strategy, the first available cloud in the clouds database is selected. We compared the performance of each cloud's response time and cost values using each selection strategy. In the remaining part of this section, we describe each scenario in greater detail.

Scenario 1: We evaluated the CWTS algorithm against three other single-strategy trust score evaluation algorithms. We ran our simulation using the CWTS algorithm with equal weight values for each of the three strategies, and ran each strategy separately, as explained in the previous section. We measured the total data processing cost and response time of the selected cloud using each algorithm. We first ran this test using data populated by our simulation framework, followed by CloudSim-generated data. The results are shown in FIGURE 11 by two graphs indicating that the CWTS algorithm yields an average time response and cost compared with the other algorithms, which is due to using equivalent weights for each algorithm for evaluating the trust score of the CWTS.

Scenario 2: We changed the weight assigned to each strategy for trust score evaluation using the CWTS algorithm

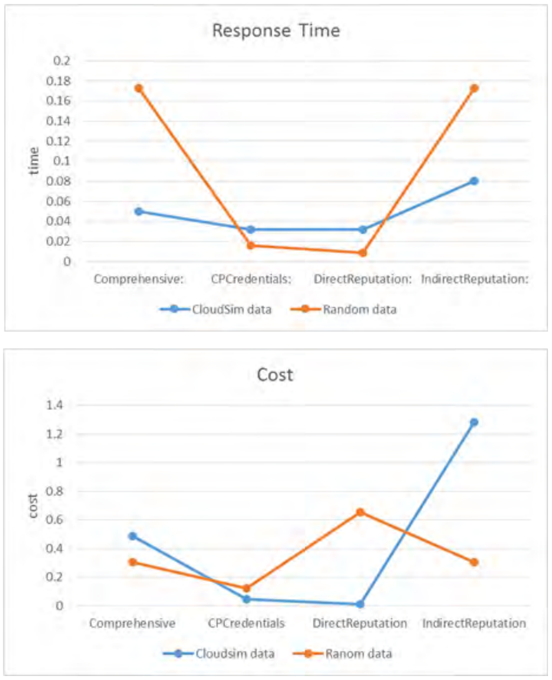


FIGURE 11. **Response time and cost of the CWTS vs. other strategies.**

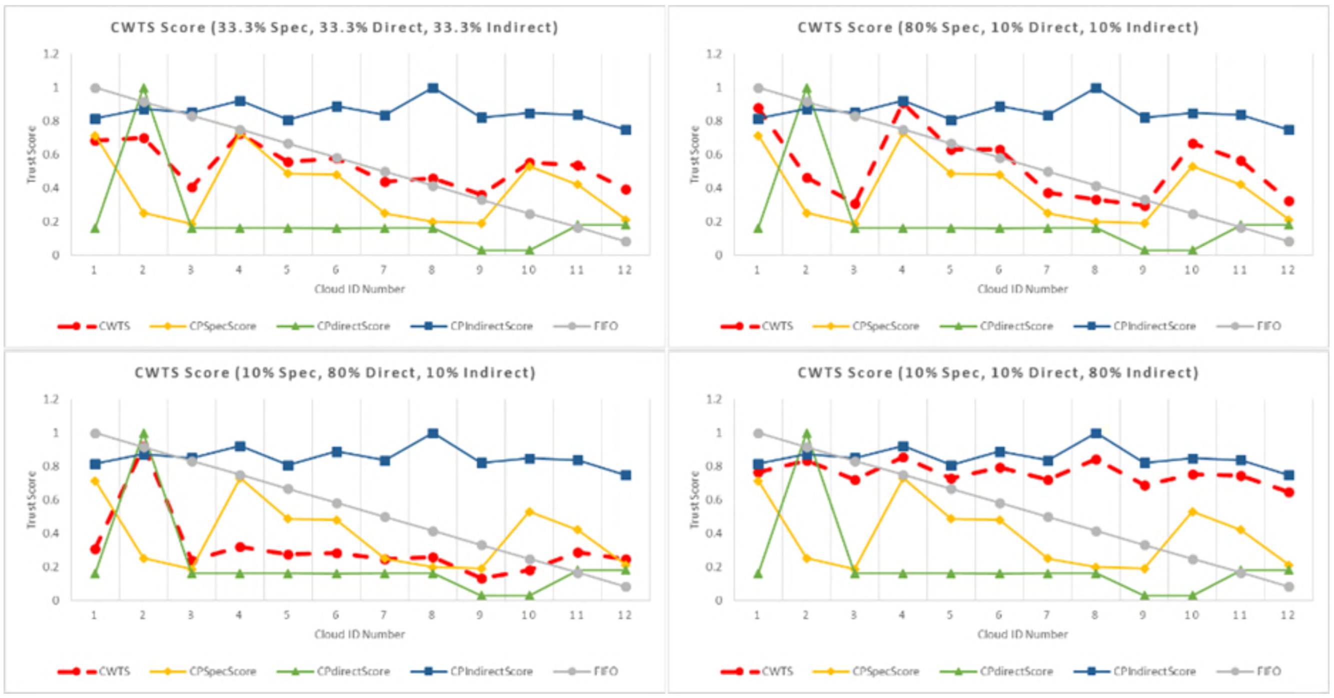


FIGURE 12. **Behavior of CWTS under various assigned weight values for each strategy.**

and compared the resulting trust score values with those calculated via the *Cloud_Spec*, *Direct_Rep*, *Indirect_Rep* and FIFO strategies. FIGURE 12 shows the trust score of all cloud providers for each selection strategy. In this graph, we notice a trend in the CWTS algorithm; it adapts to the changes in behaves similarly to the strategy with the highest assigned weight. weights assigned to each strategy, and the CWTS graph line. In addition, the FIFO algorithm graph line tends to decrease because it does not consider reputation, cloud characteristics or quality attributes. It assigns the highest weight to the first available cloud provider followed by gradually decreasing weights for the remaining cloud providers.

Scenario 3: We evaluated the performance of our CWTS algorithm in the presence of malicious or false reputation information from neighboring users. We measured the trust score of the selected cloud using the *CWTS*, *Cloud_Spec, Direct_Rep* and *Indirect_Rep* algorithms against the percentage of false ratings.

In FIGURE 13, the trust score value for the *Indirect_Rep* strategy decreases dramatically as the percentage of false ratings increases, which proves that depending solely on indirect reputation is unconstructive for selecting a suitable cloud. It also shows that the trust score generated by the CWTS algorithm is resistant to false ratings as it is not significantly affected by increases in the number of false ratings. We also used the CWTS algorithm with different weight values for each strategy, giving more weight to indirect reputation, specifically; however, FIGURE 13 shows that the overall trust score of the CWTS was not affected by an increase in the false rating percentage, even when the indirect reputation strategy was given greater importance. Here, the weights are 60%, 20% and 20% for the *Indirect_Rep*, *Direct_Rep* and *Cloud_Spec* strategies, respectively. On another note, the graph lines in FIGURE 13 are based on pre-normalized trust scores; however, the CWTS score is normalized.

Scenario 4: We simulated a new user trying to choose a suitable CSP and assumed that he/she had no prior experience with any of the cloud service providers (the direct reputation database contained no data). This user relied mainly on the indirect reputation database populated by reputation information from neighbors in the community as well as the cloud specification database populated by information collected from the existing clouds registered with the community. FIGURE 14 shows that the response time and cost of querying the selected cloud were satisfactory anunaffected by the lack of direct reputation. In addition, trust scores could not have been generated for any of the cloud providers if the user had relied solely on direct reputation.

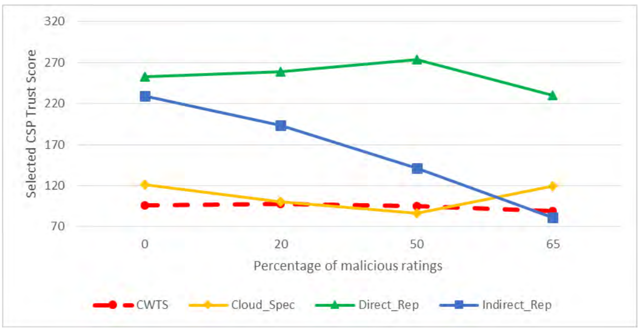


FIGURE 13. **CWTS response to false ratings.**

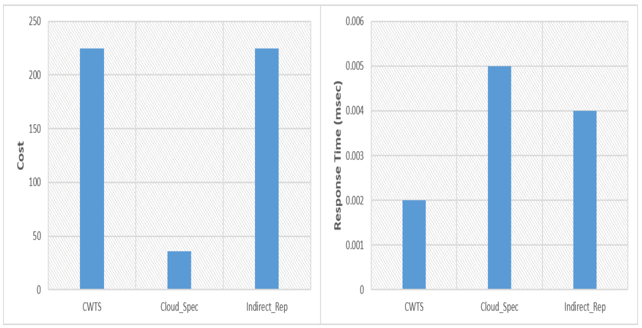


FIGURE 14. Behavior of CWTS with no user previous experience with the cloud provider.

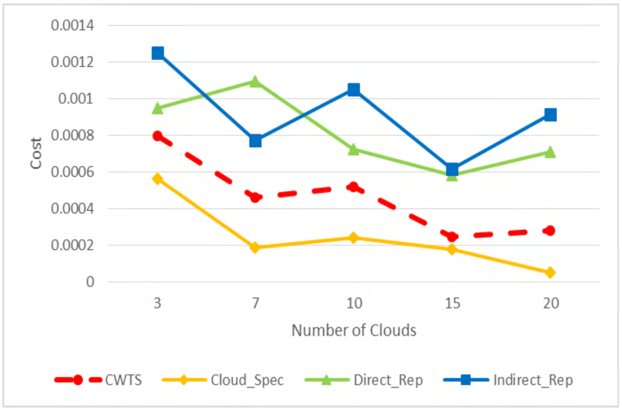


FIGURE 15. Cost measurements with an increasing number of cloud providers.

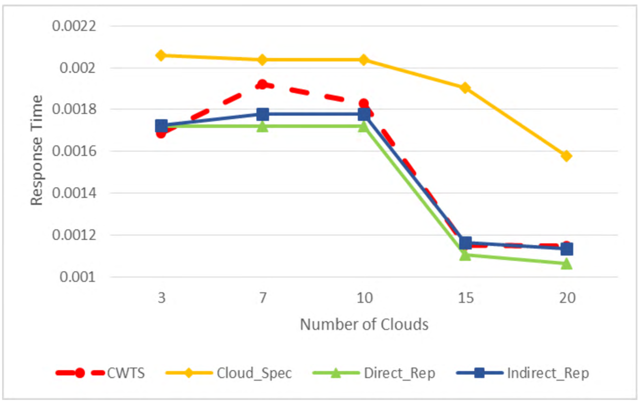


FIGURE 16. Response time measurements with an increasing number of cloud providers.

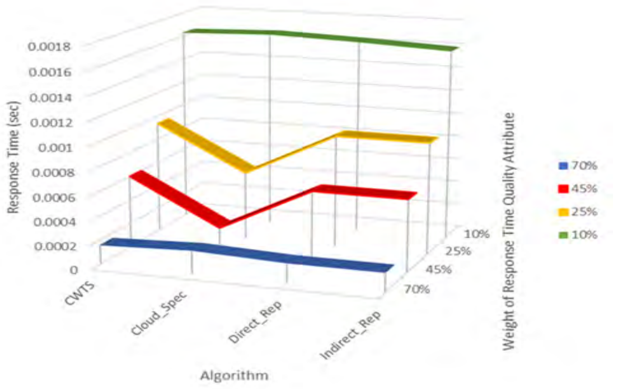


FIGURE 17. Behavior of the CWTS algorithm with various QoCS attribute weight values.

Scenario 5: We changed the number of clouds and tested whether our CWTS algorithm had better performance using a greater number of cloud providers. FIGURE 15 and FIGURE 16 show that our CWTS algorithm led to improved response times with an in increase in the number of clouds, i.e., increased options. In addition, the figure shows that lower cost is associated with an increase in the number of clouds. In this scenario, we populated our databases with randomly generated data from our simulator rather than CloudSim-generated data.

Scenario 6: We changed the weight value for a single QoCS attribute and tested the effect on the selection of the CWTS algorithm and the remaining strategies. We chose the response time as our test QoCS attribute. In this scenario, we used a random data generator to populate our databases. FIGURE 17 shows that response time decreased as its weight value increased. This can be explained by the fact that the response time was made relatively more important; thus, response time was favored over the other QoCS attributes during cloud selection.

C. OVERALL DISCUSSION AND FUTURE WORK

In this section, we discuss and evaluate our overall experimental results and suggest future plans to extend our simulations. Based on experimental and formal evaluations, in addition to the overhead evaluation described in the previous sections, we found that our trust model is sufficiently comprehensive and exhibits the following characteristics:

- It scales very well with increasingly high numbers of cloud providers, users, neighbors and QoCS properties.
- It balances the trust ratings of different stakeholders (cloud providers, customers and community members), objectively resulting in an accurate trust evaluation.
- It reacts efficiently to false ratings provided by malicious neighbors.
- It considers the user's QoCS requirements and Big Data characteristics when evaluating the trust of cloud providers.
- It regulates and control user behavior within a community of users wherein a set of rules, obligations, and penalties are enforced. This guarantees the accuracy of the ratings that contribute to the trust calculation.
- It exhibits negligible communication overhead during trust evaluation.
- It provides an application to accurately and easily retrieve the client's QoCS requirements and Big Data quality properties prior to trust evaluation.

VII. CONCLUSION AND FURTHER DISCUSSION

Over the past few years, Big Data has attracted the attention of both academia and industry. Business organizations using Big Data to get timely meaningful information definitely increase their chances of acquiring more customers, improving their operations, decreasing their costs and eventually enabling greater profits. Yet, Big Data processing is still a challenging and time consuming task that requires sizeable computational resources. Cloud computing addresses these challenges by offering resources upon request with costs that are relative to actual usage. Moreover, it allows infrastructure scalability according to dynamically changing demand.

Big Data processing, storage and distribution on the cloud remain enormously challenging issues. For example, it is extremely difficult to consolidate the contradictory quality requirements of Big Data client applications and cloud providers. On one hand, selecting the most suitable cloud to handle Big Data applications is an open issue that has not been thoroughly addressed in the literature. Different users have various levels of required and expected QoCS performance. On the other hand, cloud providers may provide inaccurate resource information for marketing purposes. To solve this issue, we based our suggested proposals on community reputation measurements.

In this paper, we have proposed a comprehensive and multi-dimensional trust evaluation model to address these challenges and guarantee proven QoCS. Our solution uses a weighted average of three cloud selection strategies: current cloud characteristics, reputation and supported historical communications. Additionally, our trust model supports multiple crucial functional and nonfunctional requirements that guarantee reliable trust evaluation.

We have also implemented a complete architecture that calculates trust scores for given cloud providers and generates a recommendation of the cloud provider with the highest trust score.

We have conducted a series of experiments, and the results prove that our trust evaluation algorithms scale well with the number of requests with varying QoCS preferences.

We have also proven that our trust model appropriately handles malicious trust scores from neighbors. The communication overhead of our solution was found to exhibit a small overhead. We have evaluated our CWTS algorithm against other strategies, and the results have convincingly shown that our CWTS algorithm selects the cloud that best matches the customer's QoCS priority requirements.

As future work we are planning to extend our CSP selection approach to combine both monitoring and prediction techniques to continuously update the CSP ranking for efficient selection. To handle the CSP selection problem among a very large number of cloud providers, we intend to partition and distribute largescale selection matrices generated by selection algorithms on a MapReduce platform, this will save efficiently time and resources.

REFERENCES

[1] M. D. Assuncao, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, "Big data computing and clouds: Trends and future directions," *J. Parallel Distrib. Comput.*, vols. 79–80, pp. 3–15, May 2014.

[2] M. D. Assuncao, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, "Big data computing and clouds: Trends and future directions," *J. Parallel Distrib. Comput.*, vols. 79–80, pp. 3–15, May 2015.

[3] *Big Data Meets Big Data Analytics*, SAS Inst., Cary, NC, USA, 2012.

[4] K. Normandeau. (Sep. 12, 2013). *Beyond Volume, Variety and Velocity is the Issue of Big Data Veracity*. Accessed: Mar. 17, 2015. [Online]. Available: http://insidebigdata.com/2013/09/12/beyond-volume-variety-velocity-issue-big-data-veracity/

[5] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of 'big data' on cloud computing," *Inf. Syst.*, vol. 47, pp. 98–115, Jan. 2015.

[6] M. Armbrust *et al.*, "Above the clouds: A Berkeley view of cloud computing," Dept. Elect. Eng. Comput. Sci., Univ. California Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009.

[7] W. Hussain, F. K. Hussain, and O. K. Hussain, "Maintaining trust in cloud computing through SLA monitoring," in *Proc. Int. Conf. Neural Inf. Process.*, 2014, pp. 690–697.

[8] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2008.

[9] M. Alhamad, T. Dillon, and E. Chang, "SLA-based trust model for cloud computing," in *Proc. IEEE 13th Int. Conf. Netw.-Based Inf. Syst. (NBiS)*, Sep. 2010, pp. 321–324.

[10] A. Al Falasi, M. A. Serhani, and R. Dssouli, "A model for multi-levels SLA monitoring in federated cloud environment," in *Proc. IEEE 10th Int. Conf. Auto. Trusted Comput. Ubiquitous Intell. Comput. (UIC/ATC)*, Dec. 2013, pp. 363–370.

[11] F. Jrad, J. Tao, and A. Streit, "SLA based service brokering in intercloud environments," in *Proc. CLOSER*, 2012, pp. 76–81.

[12] J. Lee, J. Kim, D.-J. Kang, N. Kim, and S. Jung, "Cloud service broker portal: Main entry point for multi-cloud service providers and consumers," in *Proc. IEEE 16th Int. Conf. Adv. Commun. Technol.*, Feb. 2014, pp. 1108–1112.

[13] R. Manikandan and G. Kousalya, "A framework for an intelligent broker model of cloud service selection," *Asian J. Inf. Technol.*, vol. 15, no. 11, pp. 1776–1784, 2016.

[14] J.-B. Wu, "A trust evaluation model for Web service with domain distinction," *Int. J. Granular Comput., Rough Sets Intell. Syst.*, vol. 2, no. 4, pp. 273–280, 2012.

[15] T. H. Noor, Q. Z. Sheng, S. Zeadally, and J. Yu, "Trust management of services in cloud environments: Obstacles and solutions," *ACM Comput. Surv.*, vol. 46, no. 1, 2013, Art. no. 12.

[16] L. Qi, W. Dou, Y. Zhou, J. Yu, and C. Hu, "A context-aware service evaluation approach over big data for cloud applications," *IEEE Trans. Cloud Comput.*, to be published.

[17] K. Gokulnath and R. Uthariaraj, "Game theory based trust model for cloud environment," *Sci. World J.*, vol. 2015, Jul. 2015, Art. no. 709827, doi: 10.1155/2015/709827.

[18] H. Yahyaoui, "A trust-based game theoretical model for Web services collaboration," *Knowl.-Based Syst.*, vol. 27, pp. 162–169, Mar. 2012.

[19] M. M. Hassan *et al.*, "QoS and trust-aware coalition formation game in data-intensive cloud federations," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 10, pp. 2889–2905, 2015.

[20] B. Khosravifar, M. Alishahi, J. Bentahar, and P. Thiran, "A game theoretic approach for analyzing the efficiency of Web services in collaborative networks," in *Proc. IEEE Int. Conf. Services Comput.*, Jul. 2011, pp. 168–175.

[21] M. K. Muchahari and S. K. Sinha, "A new trust management architecture for cloud computing environment," in *Proc. IEEE Int. Symp. Cloud Services Comput. (ISCOS)*, Dec. 2012, pp. 136–140.

[22] H. Kim, H. Lee, W. Kim, and Y. Kim, "A trust evaluation model for QoS guarantee in cloud systems," *Int. J. Grid Distrib. Comput.*, vol. 3, no. 1, pp. 1–10, Mar. 2010.

[23] A. M. Hammadi and O. Hussain, "A framework for SLA assurance in cloud computing," in *Proc. IEEE 26th Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*, Mar. 2012, pp. 393–398.

[24] A. Hammam and S. Senbel, "A trust management system for ad-hoc mobile clouds," in *Proc. IEEE 8th Int. Conf. Comput. Eng. Syst. (ICCES)*, Nov. 2013, pp. 31–38.

[25] A. Gholami and M. G. Arani, "A trust model based on quality of service in cloud computing environment," *Int. J. Database Theory Appl.*, vol. 8, no. 5, pp. 161–170, 2015.

[26] X. Li and J. Du, "Adaptive and attribute-based trust model for service level agreement guarantee in cloud computing," *IET Inf. Secur.*, vol. 7, no. 1, pp. 39–50, Mar. 2013.

[27] H. Lin, J. Hu, J. Liu, L. Xu, and Y. Wu, "A context aware reputation mechanism for enhancing big data veracity in mobile cloud computing," in *Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Auto. Secure Comput., Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM)*, Oct. 2015, pp. 2049–2054.

[28] B. Li, L. Liao, H. Leung, and R. Song, "PHAT: A preference and honesty aware trust model for Web services," *IEEE Trans. Netw. Service Manag.*, vol. 11, no. 3, pp. 363–375, Sep. 2014.

[29] P. D. Manuel, S. T. Selvi, and M. I. A.-E. Barr, "Trust management system for grid and cloud resources," in *Proc. IEEE 1st Int. Conf. Adv. Comput.*, Dec. 2009, pp. 176–181.

[30] E. D. Canedo, R. T. de Sousa, R. de Oliveira, and F. L. L. de Mendonça, "File exchange in a private cloud supported by a trust model," in *Proc. IEEE Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, Oct. 2012, pp. 89–96.

[31] M. Tang, X. Dai, J. Liu, and J. Chen, "Towards a trust evaluation middleware for cloud service selection," *Future Gener. Comput. Syst.*, vol. 74, pp. 302–312, Sep. 2016.

[32] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social Internet of Things," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1253–1266, May 2014.

[33] M. K. Goyal, A. Aggarwal, P. Gupta, and P. Kumar, "QoS based trust management model for cloud IaaS," in *Proc. 2nd IEEE Int. Conf. Parallel Distrib. Grid Comput. (PDGC)*, Dec. 2012, pp. 843–847.

[34] J. Guo and I.-R. Chen, "A classification of trust computation models for service-oriented Internet of Things systems," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jun./Jul. 2015, pp. 324–331.

[35] Y. Wang, Y.-C. Lu, I.-R. Chen, J.-H. Cho, A. Swami, and C.-T. Lu, "LogitTrust: A logit regression-based trust model for mobile ad hoc networks," in *Proc. 6th ASE Int. Conf. Privacy, Secur., Risk Trust*, 2015, pp. 1–10.

[36] A. Kanwal, R. Masood, R. Mumtaz, and M. A. Shibl, "Taxonomy for trust models in cloud computing," *Comput. J.*, vol. 58, no. 4, pp. 601–626, Apr. 2015.

[37] P. Resnick and R. Zeckhauser, "Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system," *Econ. Internet e-Commerce*, vol. 11, no. 2, pp. 23–25, 2002.

[38] A. Josang and R. Ismail, "The beta reputation system," in *Proc. 15th Bled Electron. Commerce Conf.*, 2002, pp. 2502–2511.

[39] A. Jøsang, "A logic for uncertain probabilities," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 9, no. 3, pp. 279–311, Jun. 2001.

[40] B. Yu and M. P. Singh, "An evidential model of distributed reputation management," in *Proc. 1st Int. Joint Conf. Auton. Agents Multiagent Syst.*, vol. 1, 2002, pp. 294–301.

[41] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, and X. Wang, "TRM-IoT: A trust management model based on fuzzy reputation for Internet of Things," *Comput. Sci. Inf. Syst.*, vol. 8, no. 4, pp. 1207–1228, 2011.

[42] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems," in *Proc. 1st Int. Joint Conf. Auton. Agents Multiagent Syst.*, vol. 1, 2002, pp. 475–482.

[43] C.-N. Ziegler and G. Lausen, "Spreading activation models for trust propagation," in *Proc. IEEE Int. Conf. e-Technol., e-Commerce, e-Service (EEE)*, Mar. 2004, pp. 83–97.

[44] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," Stanford Univ., Stanford, CA, USA, Tech. Rep. SIDL-WP-1999-0120, 1999.

[45] J.-H. Cho, K. Chan, and S. Adali, "A survey on trust modeling," *ACM Comput. Surv.*, vol. 48, no. 2, 2015, Art. no. 28.

[46] X. Liu, Y. Yang, D. Yuan, G. Zhang, W. Li, and D. Cao, "A generic QoS framework for cloud workflow systems," in *Proc. IEEE 9th Int. Conf. Dependable, Auto. Secure Comput. (DASC)*, Dec. 2011, pp. 713–720.

[47] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and Web service processes," *Web Semantics, Sci., Services Agents World Wide Web*, vol. 1, no. 3, pp. 281–308, 2004.

[48] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A cost-effective strategy for intermediate data storage in scientific cloud workflow systems," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. (IPDPS)*, Apr. 2010, pp. 1–12.

[49] M. Mehdi, N. Bouguila, and J. Bentahar, "A QoS-based trust approach for service selection and composition via Bayesian networks," in *Proc. IEEE 20th Int. Conf. Web Services (ICWS)*, Jun./Jul. 2013, pp. 211–218.

[50] D. He, Z. Peng, L. Hong, and Y. Zhang, "A social reputation management for Web communities," in *Proc. Int. Conf. Web-Age Inf. Manage.*, 2011, pp. 167–174.

[51] A. Gutowska and A. Sloane, "Evaluation of reputation metric for the B2C e-commerce reputation system," in *Proc. WEBIST*, 2009, pp. 1–10.

[52] B. Yu and M. P. Singh, "Distributed reputation management for electronic commerce," *Comput. Intell.*, vol. 18, no. 4, pp. 535–549, 2002.

[53] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Syst.*, vol. 43, no. 2, pp. 618–644, 2007.

[54] (2016). *Calculating a Weighted Average*. Accessed: Sep. 24, 2016. [Online]. Available: http://www.blacksdomain.com/files/Notes/Calculating_WA.php

[55] A. Adriyendi, "Multi-attribute decision making using simple additive weighting and weighted product in food choice," *Int. J. Inf. Eng. Electron. Bus.*, vol. 6, pp. 8–14, Nov. 2015. [Online]. Available: http://www.mecs-press.org/, doi: 10.5815/ijieeb.2015.06.02.

[56] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.

HADEEL T. EL KASSABI received the B.Sc. degree in computer science from The American University in Cairo in 1996 and the M.S. degree in computer science from Carleton University, Ottawa, in 2003. She defended the Ph.D. thesis from Concordia Institute for Information Systems Engineering, Concordia University, in 2018. Her current interests include big data, cloud computing, trust modeling, and data quality.

MOHAMED ADEL SERHANI received the Ph.D. degree in computer engineering from Concordia University, Canada, in 2006, and the M.Sc. degree in software engineering from the University of Montreal, Canada, in 2002. He is currently an Associate Professor with the College of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates. He is also an Adjunct Faculty with the Concordia Institute for Information Systems Engineering, Concordia University. He has published around 90 refereed publications including conferences, journals, a book, and book chapters. His research interests include cloud for data intensive e-health applications, and services, SLA enforcement in cloud data centers, and big data value chain, cloud federation and monitoring, non-invasive smart health monitoring, management of communities of web services, and web services applications and security. He has a large experience earned throughout his involvement and management of different research and development projects. He served on several organizing and Technical Program Committees and he was the Program Co-Chair of the IEEE Conference on Innovations in Information Technology (IIT'13), the Chair of the IEEE Workshop on Web Service (IWCMC'13) and the IEEE Workshop on Web, Mobile, and Cloud Services (IWCMC'12), and the Co-Chair of the International Workshop on Wireless Sensor Networks and their Applications (NDT'12).

RACHIDA DSSOULI received the master's and Diplome d'études Approfondies degrees and Doctorat de 3eme Cycle in networking from Université Paul Sabatier, Toulouse, France, in 1978, 1979, and 1981, respectively, and the Ph.D. degree in computer science from the Université de Montréal, Canada, in 1987. She is a Full Professor and the Director of the Faculty of Engineering and Computer Science, Concordia Institute for Information Systems Engineering (CIISE), Concordia University. Her research interests are in communication software engineering a sub discipline of software engineering. Her contributions are in testing based on formal methods, requirements engineering, systems engineering, telecommunication service engineering, and quality of service. She has published over 200 papers in journals and referred conferences in her area of research. She supervised/ co-supervised over 50 graduate students among them 20 Ph.D. students. She was the Founding Director of CIISE in 2002. The Institute host currently over 550 graduate students and 20 faculty members, four master programs, and a Ph.D. program.

BOUALEM BENATALLAH (M'02) received the Ph.D. degree in computer science from Grenoble University, France. He is a Scientia Professor with the University of New South Wales, Australia. His research interests include web services composition, cloud services orchestration, quality control in crowdsourcing, cognitive services, and data curation. He is a member of ACM.