Cluster Comput (2015) 18:817–828
DOI 10.1007/s10586-014-0416-6

Scheduling of big data applications on distributed cloud based on QoS parameters

Rajinder Sandhu · Sandeep K. Sood

Received: 13 June 2014 / Revised: 24 October 2014 / Accepted: 13 December 2014 / Published online: 21 December 2014

Abstract Big data is one of the major technology usages for business operations in today's competitive market. It provides organizations a powerful tool to analyze large unstructured data to make useful decisions. Result quality, time, and price associated with big data analytics are very important aspects for its success. Selection of appropriate cloud infrastructure at coarse and fine grained level will ensure better results. In this paper, a global architecture is proposed for QoS based scheduling for big data application to distributed cloud datacenter at two levels which are coarse grained and fine grained. At coarse grain level, appropriate local datacenter is selected based on network distance between user and datacenter, network throughput and total available resources using adaptive K nearest neighbor algorithm. At fine grained level, probability triplet (C, I, M) is predicted using naïve Bayes algorithm which provides probability of new application to fall in compute intensive (C), input/output intensive (I) and memory intensive (M) categories. Each datacenter is transformed into a pool of virtual clusters capable of executing specific category of jobs with specific (C, I, M) requirements using self organized maps. Novelty of study is to represent whole datacenter resources in a predefined topological ordering and executing new incoming jobs in their respective predefined virtual clusters based on their respective QoS requirements. Proposed architecture is tested on three different Amazon EMR datacenters for resource utilization, waiting time, availability, response time and estimated time to complete the job. Results indicated better QoS achievement and 33.15 % cost gain of the proposed architecture over traditional Amazon methods.

Keywords Big data · Cloud computing · Quality of Service (QoS) · Hadoop · Self organizing maps · K nearest neighbor

R. Sandhu (✉) · S. K. Sood
Guru Nanak Dev University, Regional Campus, Gurdaspur, India
e-mail: rajsandhu1989@gmail.com

S. K. Sood
e-mail: san1198@gmail.com

1 Introduction

Cloud computing is most discussed and promising topic of information technology field in recent times. It is listed in Gartner top ten technologies list for last three consecutive years [1]. Organizations that require dynamic information technology infrastructure are moving to cloud due to its scalability and effective pricing models. Cloud provides a wide variety of services to its end users such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and now a days it is called Anything as a Service (XaaS). Despite achieving the recognition and popularity, cloud computing faces some of the key challenges in its complete acceptance by information technology and communication communities [2]. Providing agreed Quality of Service (QoS) requirements to end user is one of the major challenges to cloud service providers. Cloud services provided to end user executes on third party cloud information technology infrastructure composed of distributed cloud datacenters which are located at different geographical locations. Cloud user uses these services using pay as use pricing model through internet. So, if a cloud user is not getting desired QoS, his/her work will be delayed which increases cloud usage cost and affects cloud user faith in future adoption for cloud services.

With the increase in the amount of data and incapability of traditional databases to process unstructured data, big-data analytics have gained new heights in this era. Every organization wants to process and extract useful information from a vast set of accumulated raw data of multiple formats [3]. Cloud computing can offers an effective environment for big data analytics applications. But, timely and correct information extraction in big-data analytics is very important. So, cloud computing should provide and maintain certain QoS to guarantee user's desired result in time. For the same amount of data, different cloud users will observe different QoS from same cloud service provider due to difference in their network infrastructure capabilities. QoS parameters for big data applications are not only dependent on the cloud service provider but on cloud user resources also. For instance, big data requires large data transfer and if any cloud service provider delivers bandwidth of 5 GB/s but the cloud user has network capability of 1 GB/s then the cloud user will not achieve QoS as advertised by cloud service provider. Cloud consists of large datacenters located at different geographical locations around the world. These distributed datacenters can be used by same cloud service provider for providing same cloud services to multiple users. So, QoS based scheduling algorithms for single datacenter are not efficient for the global perspective of big data and cloud computing. Single datacenter based algorithms do not consider location and network traffic latencies of different geographical distributed datacenters. Scheduling big data applications on multiple datacenters based on QoS parameters is very important aspect and requires special attention. Many researchers provided scheduling algorithms based on cloud service provider's QoS parameters and for local datacenter level. However, allocating and managing new incoming job to specific datacenter at geographical level and at specific virtual cluster by a single architecture can solve some of critical issues. So, the aim of study presented in this paper is to schedule big data application to geographical distributed cloud datacenters based on QoS parameters at both fine and coarse grained level which are datacenters and clusters respectively.

To achieve the suggested aim, an architecture is proposed to schedule big data application requests to appropriate datacenter at a coarse grained level and to efficient virtual cluster at a fine grained level using the global scheduler and local scheduler respectively. Adaptive K-nearest neighbor (AKNN) algorithm in global scheduler is used to find appropriate local datacenter depending on user location and requirements. Big data request are classified into general purpose, computational intensive, memory intensive and input/output intensive based on their QoS requirements using naïve Bayes algorithm. Self Organizing Maps (SOM) technique in local datacenter is used to create virtual clusters for specific type of big data request. SOM creates a topological ordering of virtual clusters so that virtual clusters near in topological ordering are more related to each other. Each incoming big data request is allocated to its specific virtual cluster for its better performance and efficient resource scheduling. Proposed architecture is implemented on Amazon Elastic Map Reduce (EMR) and compared to traditional Amazon techniques. It provides better QoS requirements and 33.15 % cost gain over traditional Amazon methods.

Organization of rest of the paper is as follows. Section 2 investigates work related to scheduling of cloud computing resources based on QoS parameters. Section 3 proposes an architecture for scheduling of the big data request to cloud datacenters based on QoS parameters. Section 4 provides experimental setup, results and discussion. Lastly, Sect. 5 concludes the paper and provides future work.

2 Related work

This section investigates work done in the field of QoS based cloud resource scheduling for different cloud applications such as cloud ranking, cloud multimedia streaming, mobile cloud processing and cloud gaming.

In 2013, Zhang et al. [4] proposed a framework for ranking of cloud service providers on the basis of QoS provided by them. They considered that same cloud service provider will provide different QoS to different cloud users depending on particular cloud user's resources capabilities. Two ranking algorithms which are CloudRank1 and CloudRank2 are described which uses QoS parameters to rank different cloud service providers. In 2013, Rao et al. [5] developed a two layer QoS provisioning framework known as DynaQoS based on Self-Tuning Fuzzy Control (STFC). They classified cloud service providers in basic and premium class based on their QoS parameters. They implemented it on the Xen based cloud testbed. In 2013, Wang et al. [6] proposed an adaptive scheduling algorithm for scheduling jobs over a hybrid cloud by maintaining desired QoS. They argued that to maintain QoS in a hybrid cloud, private cloud resources should be maximally utilized which will also reduce cost for public cloud usage. They cut the number of tasks executing on public cloud by shifting them to private cloud according to required QoS parameters. They showed that their algorithm achieved better private cloud resource utilization and had a higher QoS satisfaction rate as compared with FIFO and FAIR scheduler inbuilt in CloudSim 2.1.1.

In 2013, Zhu et al. [7] provided a strategy to increase QoS of video streaming over the heterogeneous cloud systems and termed it as cloud assisted Scalable Video Coding (SVC) streaming. They proposed to adapt cloud network resources at network level to maintain standard quality of video streaming. Similarly, in 2014, Hsu and Lo [8] mapped QoS parameters of cloud network to Quality of

Experience (QoE) parameters of users for multimedia applications. They provided a QoE function using eight different QoE parameters. Every video streaming is relayed with guaranteed bandwidth to every user. After the video, a QoE score is calculated. If QoE score goes below a threshold value, it is assumed that QoS parameters are not achieved by cloud.

In 2013, Lin et al. [9] proposed two separate algorithms for efficient data replication for data intensive applications. A greedy high QoS first replication algorithm provides effective data replication, but the cost of replication was very high. Second algorithm which was based on minimum cost and maximum flow produces an optimal solution in polynomial time but require high computation power. They argued to perform effective tradeoff between both the algorithms.

In 2013, Misra et al. [10] studied the issue of maintaining QoS in mobile cloud environment due to mobility of mobile user. They achieved the desired QoS for each mobile user by shifting the bandwidth to required mobile user using a modified bid auction algorithm termed it as Auction-based QoS-guaranteed Utility Maximization (AQUM). They showed that by using their algorithm, mobile device achieved Nash equilibrium.

In 2014, Chen et al. [11] discussed the importance of QoS in the cloud gaming environment. They established a measurement technique for evaluating different cloud gaming systems provided by separate organizations based on user expected QoS. They compared two cloud gaming systems OnLive and StreamMyGame by using the measurement technique and suggested that OnLive provides better QoS to users.

In 2014, Kaur and Chana [12] proposed a framework for multi-user aware execution of multi-tier web applications hosted on the cloud. They studied the behavior of the web application and predict the amount of resources required to attain scalability as well as achieve desired QoS parameters. QoS mapper is used to analyze web application as well as resource utilization data. They implemented it on Amazon EC2 for a cloud health application.

3 Proposed framework

Scheduling of cloud resources according to QoS parameters is very important for big data applications. Figure 1 shows the proposed framework for the scheduling of big data applications over geographically distributed cloud datacenters. Two different schedulers are used for efficient scheduling of cloud resources. Global scheduler is used at a coarse grain level and local scheduler is used at a fine grained level. These schedulers are responsible for selecting appropriate datacenter and cluster for a big data application request. Further sub sections describe both schedulers in detail.

3.1 Global scheduler

Multiple functional and quality attributes are associated with any cloud datacenter and big data processing request from any user. As the amount of data in big data applications is very large, so the datacenter selection decision at coarse grained level should consider parameters such as physical distance, average resources required by application, and network throughput (GB/s). Selection of very high datacenter network throughput is not useful if it is not compatible with user's network. For big data applications, three parameters are considered to select local datacenter. These three parameters are physical distance, network throughput, and available resources. Due to involvement of large amount of transferred data, physical distance and network throughput are considered in the selection of local datacenter. Even if physical distance and network throughput are desirable, local datacenter with no free resource will not provide desired QoS requirements. So, total required resources by user's application and total free resources in local datacenter are also compared.

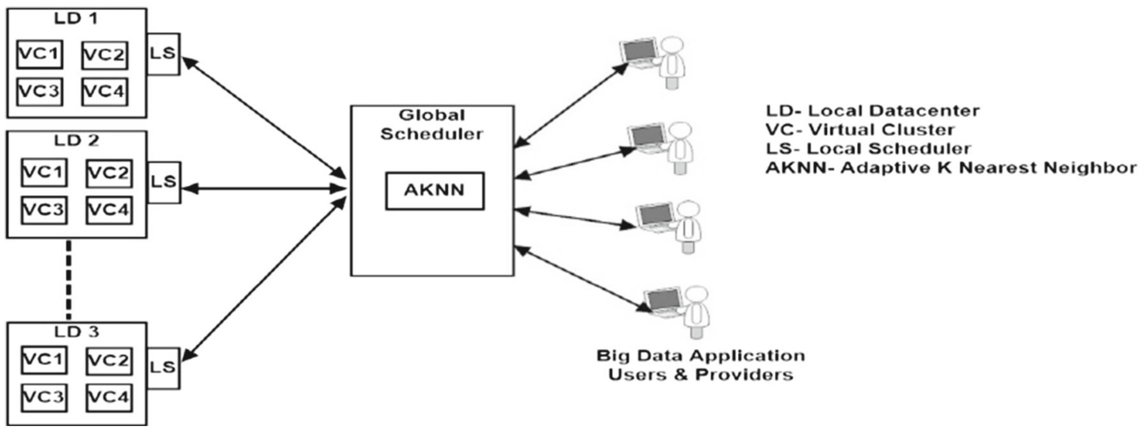


Fig. 1 Proposed framework

User's request provides parameters to global scheduler which uses AKNN algorithm to find suitable datacenter. AKNN algorithm represents each datacenter as a point in n-dimensional space where n is the number of attribute used to find the nearest neighbor. Each new incoming big data request is also represented as a point in KNN dimensional space and its distance is calculated from all other points using Euclidean distance. Physical distance and network throughput provided by any datacenter are fixed but current available resources are dynamic and vary rapidly. Simple KNN algorithm just compares static points in n dimensional space, which will be not suited for proposed architecture. Load of datacenter is dynamic in nature so it cannot be represented as a point in KNN dimensional space. To make KNN adaptive with varying load of local datacenter, AKNN compares load of big data user request with current load of selected datacenter at any kth level for each cycle. If the current load of the datacenter is smaller than big data user request, then that particular datacenter is allocated to big data user otherwise new nearest neighbor is identified by increasing the value of k, as shown in Algorithm 1. This will help in achieving QoS in big data applications at global level. This algorithm is adaptive in the sense that it compares current load of datacenter along with other functionality and QoS parameters. Algorithm 1 below explains AKNN.

U is a vector which stores two parameter values which are physical distance and network throughput.

V is a vector containing two values for a datacenter which are physical distance from requested user and network throughput provided.

D is a scalar quantity which represents selected datacenter.

k is the sequence number of nearest neighbor to user request in two dimensional space.

current_free is the amount of free resources in local datacenter at the moment of selection.

user_load is the amount of resources that big data request will consume.

3.2 Local scheduler

After the selection of local datacenter by global scheduler, big data application arrives at local scheduler. Every new big data application has different architecture so it requires different set of computational hardware. For example, computational requirement of TeraSort and MRBench Hadoop applications are different from each other. TeraSort requires more memory whereas MRBench requires more CPU cycles. So, each new big data requests arrived in proposed architecture are categories into four main categories which are general purpose, computational intensive, input/output intensive and memory intensive. Table 1 shows different cluster types and their requirements.

Each local datacenter has clusters dedicated for each category of big data request. Creating physical clusters are difficult and will reduce utilization of whole datacenter. In proposed architecture, virtual clusters are proposed which are created dynamically using a self-organizing maps algorithm. Before submitting new big data request to virtual cluster, its category will be identified. Naïve Bayes algorithm is used to probabilistically categories new request to a specified category.

3.2.1 Naïve Bayes algorithm

Category of big data application is not known to end user but he/she is roughly aware of some important functionality and QoS requirements of application. Users just have to mention some important functionality and QoS parameters associated with their application. These parameters can also be captured form SLA signed between cloud service provider and end user. Naïve Bayes algorithm used in proposed architecture is responsible for predicting category of user's big data application. Naïve Bayes predicts the probabilities using Bayes theorem which provides conditional probability formula as

Algorithm 1: Adaptive K Nearest Neighbor

```
Inputs: V, U, current_free, user_load
Output: D
Step 1: Set k = 1 and U = ($u_1$, $u_2$) for current user
Step 2: Compute the distance between user request parameters U and every datacenter parameters V = ($v_1$, $v_2$)
          $d(U,V) = |U - V| = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$
Step 3: Select $V_i \subseteq V$, the set of k closest points to U arranged in a sequence.
Step 4: for each kth nearest data center selected, calculate current free resources.
Step 5: if ( current_free < user_load)
            Allocate U to selected local datacenter D.
        Else
            k = k + 1 and goto Step 4.
Step 6: END
```

Table 1 Different cluster type requirements and applications

Cluster type	CPU performance level	Memory size level	Storage size level	Applications
General Purpose	Normal	Normal	Normal	Development environment
				Code repository
				Low traffic web application
				Small databases
Compute Intensive	High	Normal	Normal	High performance application
				Large traffic web applications
				Large analytics
				Batch processing
				Video encoding
Memory Intensive	Normal	High	Normal	High performance database
				Distributed memory cache
				In-memory data analytics
				Large genome assembly
				Deployment of SAP
				SharePoint
Storage Intensive	Normal	Normal	High	Very large transactional database
				Data warehousing
				Cluster file systems
				Distributed file system web applications

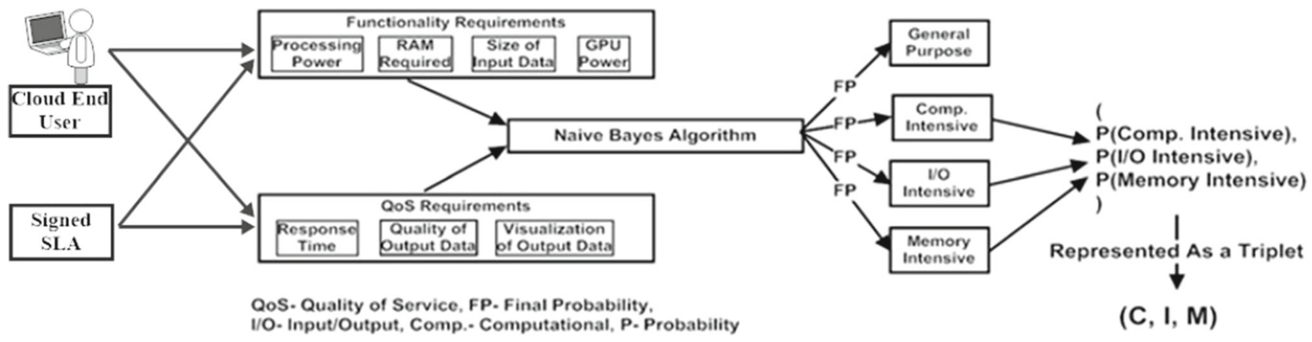


Fig. 2 Selection of category using functionality and QoS parameters

$$P(C_i|Y) = \frac{P(Y|C_i)P(C_i)}{P(Y)}$$

where, C_i is the ith output class and Y is the attribute tuple.

Category of incoming request cannot be fixed to a certain category. New incoming request may require combination of both memory and CPU cycles. So, a probabilistic association of incoming requests to categories is required in proposed architecture. Naïve Bayes classifier finds the probability of each user request to fall in any of four specified categories of big data and creates a triplet (C, I, M) where C, I and M shows the probability of incoming request to fall in computational intensive, input/output intensive or memory intensive respectively. By default, the job is assumed to fall in general purpose cluster of big data datacenter. Figure 2 shows functionality and QoS parameter linking with categories and generation of (C, I, M) triplet using naïve Bayes algorithm. Training data is collected from different configurations of Amazon EMR [13], Rackspace [14] servers. Table 2 shows a set of different attributes collected for Amazon and Rackspace cloud service providers for training and testing of naïve Bayes algorithm.

The tenfold cross validation training and testing process is done on data collected for predicting the accuracy of naïve Bayes classification in proposed architecture. Weka 3.7 [15] is used to test the use of naïve Bayes algorithm to find appropriate category of big data jobs. Table 3 shows summary of naïve Bayes in weka 3.7. On training data, naïve Bayes algorithm predicts 144 instances correctly out of 150 which provides accuracy rate of 96 %. It also provides low mean

Table 2 Set of training data for Naïve Bayes algorithm

S. No.	RAM (GB)	vCPU	System disk (GB)	Bandwidth (MB/s)	Cluster type
1	1	1	20	200	General purpose
2	7.5	4	40	400	Compute intensive
3	30.5	4	40	400	Memory intensive
4	30.5	4	400	750	Storage intensive
5	30.5	4	200	5000	Input/output intensive

Table 3 Summary of tenfold cross validation training of Naïve Bayes in Weka 3.7

Stratified cross-validation Summary		
Correctly classified instances	144	96 %
Incorrectly classified instances	6	4 %
Kappa statistic	0.91	
K&B relative info score	12832.1098	
K&B information score	219.5694 bits	1.4637 bits/instance
Class complexity | Order 0	227.7424 bits	1.5182 bits/instance
Class complexity | Scheme	27.5169 bits	0.1834 bits/instance
Complexity improvement (Sf)	206.1376 bits	1.3742 bits/instance
Mean absolute error	0.0332	
Root mean squared error	0.165	
Relative absolute error	7.8947 %	
Root relative squared error	33.8754 %	
Total number of instances	150	

absolute error rate and high kappa statistic, which shows the reliability of using naïve Bayes algorithm in proposed architecture.

3.2.2 Modified self-organizing maps (SOM)

The Kohonen Self-Organizing Maps (SOM) [16] is a prototype clustering technique based on neural network. SOM creates different reference vectors (also known as centroids) and assign data objects to a particular reference vector. It visualizes high dimensional data by mapping it graphically to lower dimensions and defining a pre-defined topological ordering relationship between all reference vectors. Dimension reduction by SOM helps artificial systems as well as users to visualize data effectively. SOM generates topological ordering using a competitive learning algorithm over unsupervised neural networks. During the training process, SOM uses winning neuron strategy to update and create a topological ordering between reference vectors. So, reference vectors that are close in topological ordering are more related to each other than others.

SOM is explained using simple color topological ordering. Every color is composed of different combination of three colors which are Red, Green and Black represented as (R, G, B). Figure 3 shows color topology reference vectors of different combination of red, green, and blue colors. Initially all three values of (R, G, B) are set equal to zero and color generated is black, left topmost corner of Figure 3. Value of (R, G, B) is change to generate new colors. All boxes near to blue box have more shade of blue than other boxes, similar for green and red box also. Boxes in the middle have some different values of (R, G, B). In this way, color boxes that are closer to each other are more related to each other according to (R, G, B) schema. User can select desired color based on his/her requirement of (R, G, B) combination. SOM topological ordering provides more efficient choices than a static model. Using same concept, all machines available in datacenter are arranged in topological order using SOM technology.

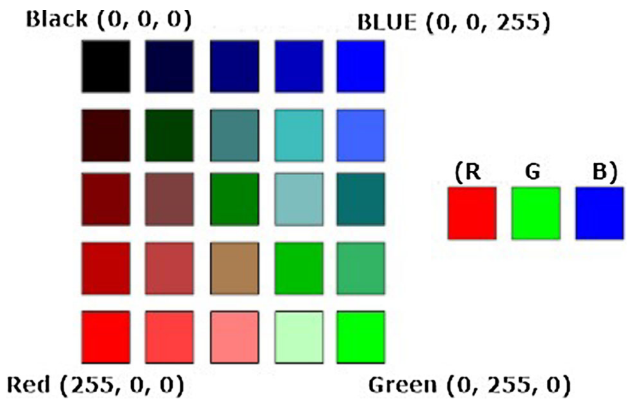


Fig. 3 Color topology using SOM

Creating physically distinguished and static clusters for each category of big data request in any datacenter is inefficient process because it is not suitable for mixed type of load and it provides low resource utilization. If any request has mixed requirement for computational as well as memory than allocating it to physically distinguish static cluster will not achieve desired QoS requirements. SOM provides an efficient method to arrange all available hardware resources in datacenter in a topological ordering. Proposed architecture uses SOM for creating virtual clusters of four types which are General Purpose, Compute Intensive, Input/output Intensive and Memory Intensive in any local datacenter. Virtual clusters are arranged in a topological ordering so that

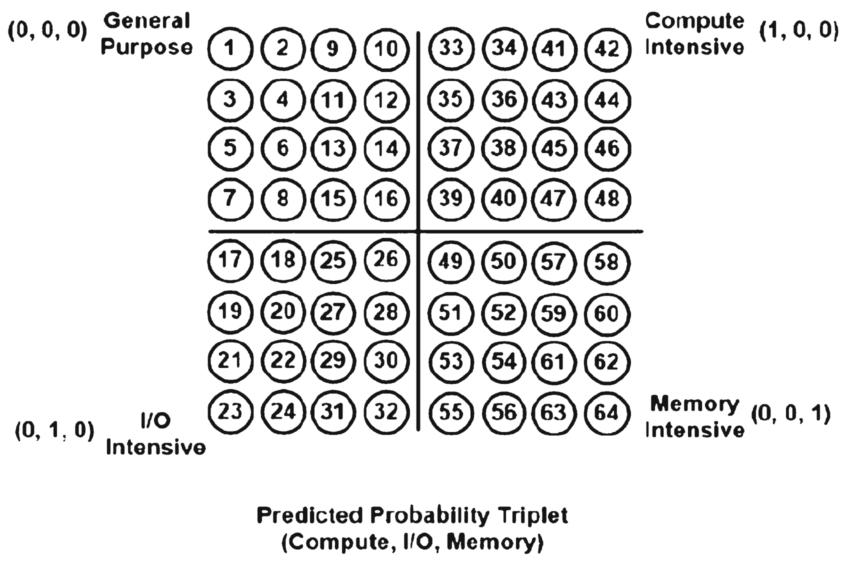
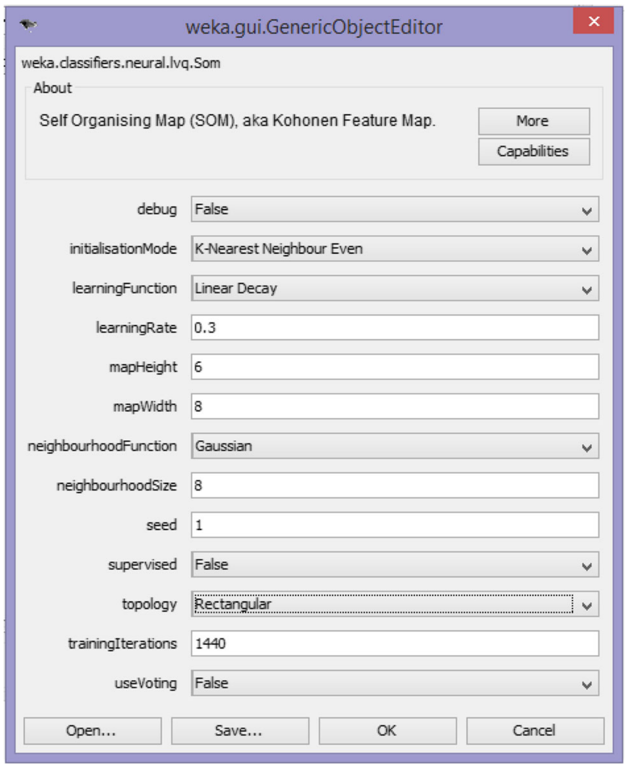


Fig. 4 Two dimensional reference vector in SOM

any type of load can be allocated to its specific virtual cluster. It makes system more dynamic in nature and adaptable to any type of request that arrives at system. Algorithm 2 explains the process of creating virtual clusters. Figure 4 shows topological ordering of any datacenter based on types of jobs classified by a triplet (C, I, M) which is predicted by naïve Bayes algorithm at local scheduler. Left topmost corner shows triplet as (0, 0, 0) which represents that naïve Bayes has predicted a zero probability of new job to fall in any of compute intensive, input/output intensive, or memory intensive category so it is a general purpose job. Right topmost corner shows triplet as (1, 0, 0) which represents that naïve Bayes predicted with certain probability that it is compute intensive job. Similarly, topological ordering is created for input/out intensive and memory intensive jobs at left lowermost corner and right lowermost corner respectively. Each created virtual cluster specification and QoS parameters will be stored at local scheduler. Every incoming request is assigned to a specific virtual cluster depending upon probability triplet generated by naïve Bayes algorithm. If specific cluster is over loaded, it can share resources from free virtual cluster closest in the topological ordering as explained in Algorithm 3. Using this approach, big data application request waiting time will be reduced, the datacenter will be fully utilized and QoS parameters will be achieved. SOM algorithm can be implemented for larger systems using MATLAB library SOM Toolbox [16] and different functions available in Weka 3.7 [15] classification tool. In this paper, Weka 3.7 is used to form cluster for different cluster types. Figure 5 shows the setting used to perform the clustering process.

category_list[] is a list of big data application categories created on the based of functionality and QoS requirements.

server_list[][] is a list containing all server machines installed in the datacenter with their specifications.



weka.classifiers.neural.lvq.Som

About

Self Organising Map (SOM), aka Kohonen Feature Map.

debug	False
initialisationMode	K-Nearest Neighbour Even
learningFunction	Linear Decay
learningRate	0.3
mapHeight	6
mapWidth	8
neighbourhoodFunction	Gaussian
neighbourhoodSize	8
seed	1
supervised	False
topology	Rectangular
trainingIterations	1440
useVoting	False

Fig. 5 Setting of Weka 3.7 for clustering of category type based on SOM

$z(t)$ is the current server machine under consideration at time t.

$m_j(t)$ is the value of closest reference vector to z(t) at time t.

$y_j(t)$ is the neighborhood effect between m_j and y at time t for creating topological ordering.

Algorithm 2: Creating Virtual Cluster using MSOM Technique

```
Input: category_list[ ], server_list[ ][ ]
Output: virtual clusters
Step 1: Set p = 0
Step 2: for all elements in category_list[ ]
        Step 2.1: Set q = 0
        Step 2.2: for all machines in server_list[ ][ ]
                Step 2.2.1: Determine the closed reference vector for server_list [q][] based on category_list [p].
                Step 2.2.2: Update the close reference vectors using
                        $m_j(t+1) = m_j(t) + y_j(t)(z(t) - m_j(t))$
                Step 2.2.3: set q = q + 1
        Step 2.3: set p = p + 1
Step 3: Assign each server machine to its closest reference vector
Step 4: Return reference vectors and created virtual clusters.
```

resource utilization table is maintained by the local scheduler about the current and past resource utilizations of clusters in the datacenter.

cluster_free is the cluster, which is near to the current cluster in topological ordering and has some free space to execute new job.

p [] is the sorted list of top three big data applications categories created based on conditional probabilities generated by naïve Bayes algorithm.

Algorithm 3: Job Assignment in Virtual Clusters

```
Inputs: Resource utilization table, current cluster, new request, p[ ].
Step 1: Set i = 0.
Step 2: Select cluster from p[i]
Step 3: if cluster is over utilized
        Step 3.1: Find a virtual cluster close in topological ordering that has free resources.
        Step 3.2: If (cluster_free ≠ null)
                Step 3.2.1: Shift desired resources from cluster_free to selected cluster.
                Step 3.2.2: Update resource utilization table.
        Step 3.3: Else
                Step 2.3.1: Set i = i + 1 and goto Step 2.
Step 4: Else
        Step 4.1: Allocate request to selected virtual cluster.
Step 5: End.
```

4 Experimental execution analysis

The proposed framework is tested for Hadoop v1.0.3 applications on Amazon Elastic Map Reduce (EMR) clusters. This section explains the experimental setup, results, and discussion.

4.1 Experimental setup

Figure 6 shows the experimental setup used to test the proposed framework. Four types of clusters are created at three different locations of Amazon Elastic Map Reduce (EMR) datacenters. Amazon EMR [13] are Hadoop clusters developed on Amazon EC2 [17] infrastructure for different map reduce tasks. Types of cluster developed are general purpose, compute optimized, input/output optimized and memory optimized. Input/output intensive cluster is created by increasing bandwidth of general purpose cluster. A local server is deployed in our university which is responsible for scheduling proposed QoS algorithm and monitor different QoS parameters of all datacenters. Four Hadoop benchmark which are TeraDFSIO (input/output intensive), TeraSort (memory intensive), NNBench (General) and MRBench (compute intensive) are used to evaluate the proposed framework. All four benchmarks send request to local server, which forward request to appropriate datacenter and cluster according to proposed architecture. Virtual Private Networks (VPNs) are used to send requests of Hadoop benchmarks from United States, Europe and India to local server located in India. New batch of job is submitted from different VPNs to local server after every 5 min for total experiment

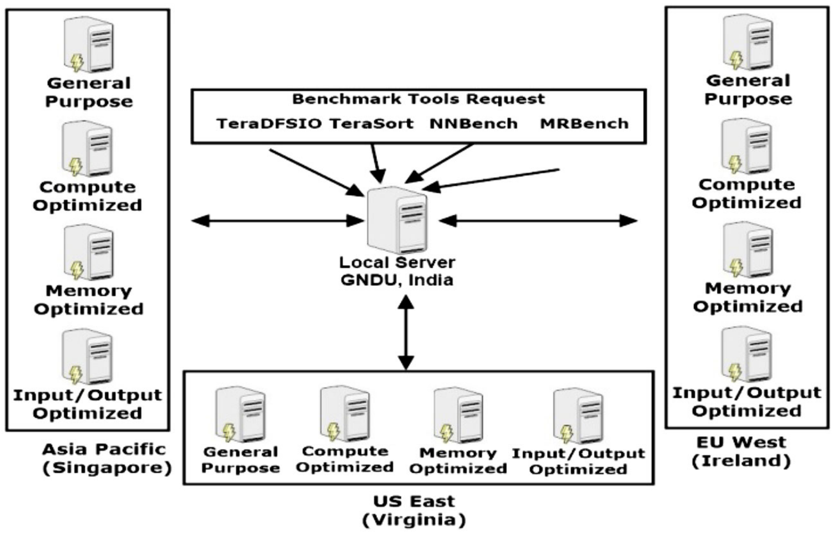


Fig. 6 Experimental setup for testing proposed framework

of 3 h. Each batch of job submitted contains all types of jobs according to a fixed percentage which is 30 % general purpose, 25 % compute intensive, 25 % memory intensive and 20 % input/output intensive. All benchmark jobs are scheduled according to proposed architecture and normal Amazon scheduling policies by local server. Normal Amazon scheduling policy does not consider network bandwidth and physical distance between requests and datacenters. It also does not categorize incoming jobs so that they can allocate to desired clusters. All incoming batch of jobs by Amazon are scheduled to only US East datacenter's general purpose clusters. Proposed architecture technique and Amazon scheduling are compared for different QoS parameters as shown in Fig. 7.

4.2 Cost analysis

Cost of whole experiment is analyzed for both proposed and Amazon scheduling policies. Table 4 shows the cost of usage of per instance of all clusters for each datacenter. Proposed architecture uses specific clusters for each request whereas Amazon uses only general purpose cluster for all requests.

Table 5 lists the values of cost for the proposed and Amazon approach. The cost is calculated for every 30 min of interval. Total cost is composed of actual usage cost and cost that will be used for completing all waiting jobs in a particular interval of time. The total cost for an interval of 30 min is calculated as follows:

$$\textit{Total cost}\ (30min) = \textit{Usage cost} + \sum_{i=0}^{n} \textit{TTC} * \textit{CPH}$$

where, *Total cost* is the cost of a particular 30 min interval. *Usage cost* is actual usage cost calculated by Amazon services for 30 min interval. *n* is total number of job waiting or running after 30 min interval ends. *Time to complete (TTC)* is the estimated time that a running job will take to complete. *Cost per hour* (CPH) is the cost of a particular job running on the cluster.

After the completion of experiment, total cost of proposed architecture is \$101.15 and traditional method is \$151.33, as shown in Table 5. Proposed method provides 33.15 % cost gain over traditional method in 3 h of experiment as calculated below [18].

$$\begin{aligned} \textit{Cost gain} &= \frac{(\textit{Amazon} - \textit{Proposed})}{\textit{Amazon}} * 100 \\ &\Rightarrow \frac{(151.33 - 101.15)}{151.33} * 100 = 33.15\,\% \end{aligned}$$

4.3 Discussion

Proposed architecture is simple yet effective to schedule big data jobs to geographically distributed datacenters. It works on simple method to first schedule job to nearest datacenter because of large amounts of data has to be transferred. Secondly, it schedules jobs to specific virtual clusters according to their type. Experimental performance evaluation of proposed architecture is done using Amazon EMR against different QoS values which are shown graphically in Fig. 7. Figure 7a represents resource utilization of all datacenters using both proposed and Amazon approach for scheduling. Proposed model has slightly higher resource utilization rate. In Amazon scheduling policy, input/output jobs can be executed on general purpose cluster, which reduces resource utilization because a job is waiting for input or output. Proposed architecture provide resources to input/output requests from input/output virtual cluster so that other jobs do not suffer which in turn increases overall resource utilization of datacenter. QoS parameters of proposed architecture are slightly better than Amazon in the initial hour of the experiment, but after approximately 110 min of experiment, difference

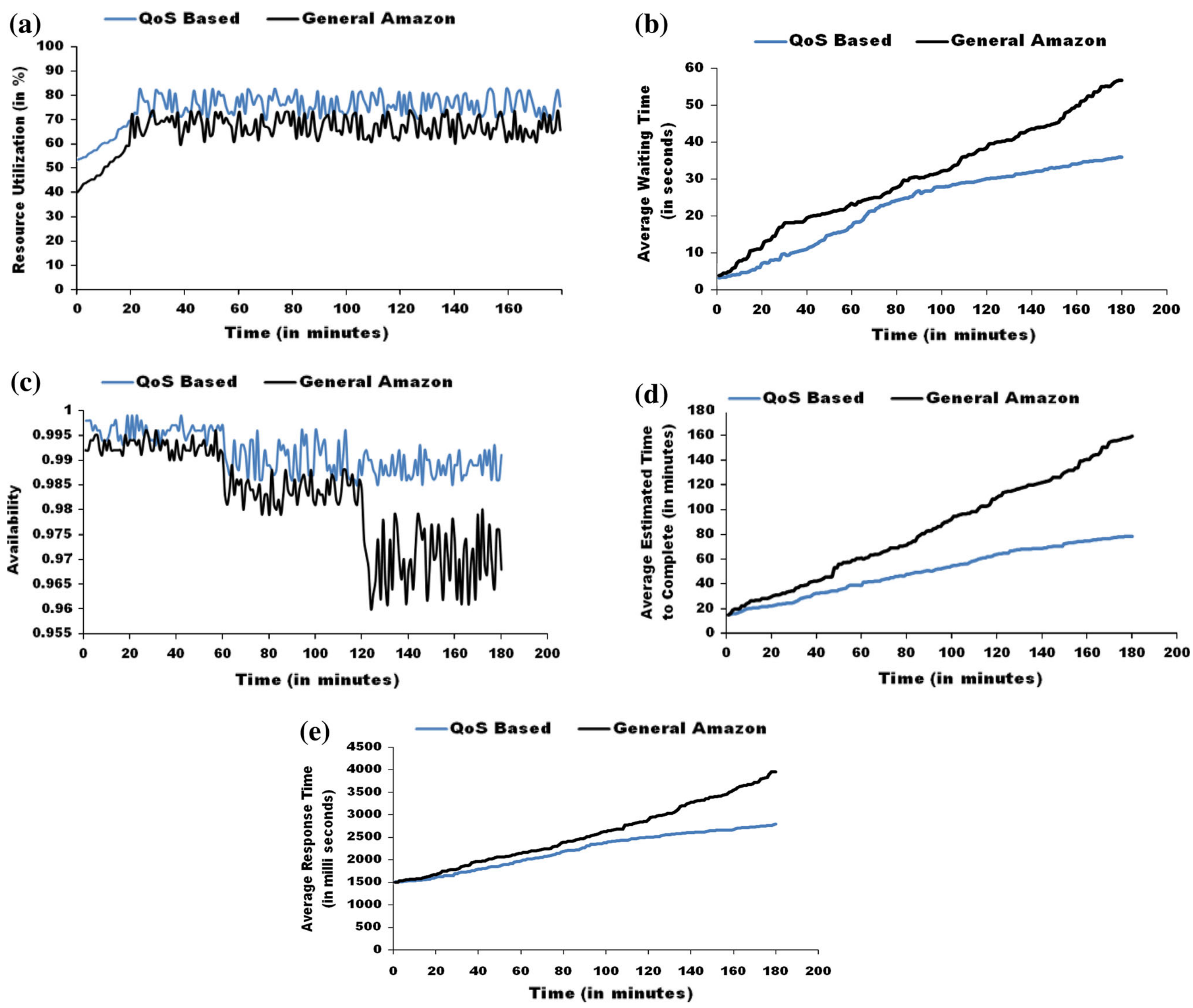


Fig. 7 Comparison of proposed architecture scheduling and simple amazon scheduling on QoS parameters of big data jobs. **a** Resource utilization of all datacenters, **b** average waiting time of jobs for start of execution, **c** availability of all datacenters, **d** average estimated time to complete of all submitted jobs, **e** average response time of all datacenters after a job start execution

Table 4 Pricing of Amazon EMR clusters [13]

Region	Cluster	Pricing (per hour)
Virginia	General Purpose (GP1)	\$0.070
	Compute Optimized (CO1)	\$0.053
	Memory Optimized (MO1)	\$0.090
	Input/output Optimized	\$0.079
Singapore	General Purpose (GP2)	\$0.070
	Compute Optimized (CO2)	\$0.053
	Memory Optimized (MO2)	\$0.090
	Input/output Optimized	\$0.079
Ireland	General Purpose (GP3)	\$0.070
	Compute Optimized (CO3)	\$0.053
	Memory Optimized (MO3)	\$0.090
	Input/output Optimized	\$0.079

started to increase linearly. The jobs are starting to complete faster in the proposed architecture as shown in Fig. 7d, it affects all other QoS parameters. There is decrease in waiting time and availability increases for all clusters as depicted in Fig. 7b, c respectively. Proposed architecture is also cost effective from Amazon scheduling policies. Cost of cloud usage will increase with increase in waiting time of arrived jobs. Considering both running and waiting jobs, proposed architecture provides 33.15 % cost gain in 3 h of experiment.

5 Conclusion

In this paper, a QoS aware big data scheduling architecture for geographically distributed cloud datacenter has been pro-

Table 5 Comparison of cost for both approaches

Time (in min)	Proposed architecture			Amazon scheduling			Total	
	DC1	DC2	DC3	DC1	DC2	DC3	Proposed	Amazon
0–30	3.42	4.12	3.13	5.75	6.04	3.92	10.67	15.71
30–60	4.51	5.98	5.12	5.90	6.18	5.68	15.61	17.76
60–90	6.12	7.41	7.99	7.20	8.53	6.66	21.52	22.39
90–120	6.01	6.87	7.11	8.55	9.99	9.26	19.99	27.8
120–150	5.77	5.03	6.53	10.28	10.98	10.49	17.33	31.75
150–180	4.43	5.59	6.01	11.12	12.24	12.56	16.03	35.92
Total	30.26	35.00	35.89	48.80	53.96	48.57	101.15	151.33

DC Datacenter and all costs are in $

posed. It is evaluated using three different geographically located Amazon EMR datacenters for several QoS parameters such as CPU utilization, waiting time, availability, estimated time to complete and response time. It investigated the impact of QoS scheduling by comparing it with the tradition Amazon model. In today's marketplace where correct, timely and efficient data extraction can affect many organizational decisions. Proposed model helps to reduce data extraction cost as well as provides desired results timely. Future work will include development of load balancer at global and local scheduler level, which will balance the load among datacenters and virtual clusters based on dynamic QoS requirements of big data application. We will also extend proposed architecture to different cloud applications such as media streaming and cloud gaming etc.

References

1. Top 10 Strategic Technology Trends of 2014 [Online]. http://www.gartner.com/technology/research/top-10-technology-trends/. Accessed 15 Oct 2014
2. Sood, S.K., Sandhu, R.: Matrix based proactive resource provisioning in mobile cloud environment. Simul. Model. Pract. Theory (2014). doi:10.1016/j.simpat.2014.06.004
3. Chen, J., Chen, Y., Du, X., Li, C., Lu, J., Zhao, S., Zhou, X.: Big data challenge: a data management perspective. Front. Comput. Sci. **7**(2), 157–164 (2013)
4. Zheng, Z., Wu, X., Zhang, Y., Lyu, M.R.: QoS ranking prediction for cloud services. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1213–1222 (2013)
5. Rao, J., Wei, Y., Gong, J., Xu, C.Z.: QoS guarantees and service differentiation for dynamic cloud applications. IEEE Trans. Netw. Serv. Manag. **10**(1), 43–55 (2013)
6. Wang, W.J., Chang, Y.S., Lo, W.T., Lee, Y.K.: Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environment. J. Supercomput. **66**(2), 783–811 (2013)
7. Zhu, Z., Li, S., Chen, X.: Design QoS-aware multi-path provisioning strategies for efficient cloud-assisted SVC video streaming to heterogeneous clients. IEEE Trans. Multimed. **15**(4), 758–768 (2013)
8. Hsu, W.H., Lo, C.H.: QoS/QoE mapping and adjustment model in the cloud-based multimedia infrastructure. IEEE Syst. J. **8**(1), 247–255 (2014)
9. Lin, J.W., Chen, C.H., Chang, M.: QoS-aware data replication for data-intensive applications in cloud computing systems. IEEE Trans. Cloud Comput. **1**(1), 101–115 (2013)
10. Misra, S., Das, S., Khatua, M., Obaidat, M.S.: QoS-guaranteed bandwidth shifting and redistribution in mobile cloud environment. IEEE Trans. Cloud Comput. **2**(2), 181–193 (2013)
11. Chen, K.T., Chang, Y.C., Hsu, H.J., Chen, D.Y., Huang, C.Y., Hsu, C.H.: On the quality of service of cloud gaming systems. IEEE Trans. Multimed. **16**(2), 480–495 (2014)
12. Kaur, P.D., Chana, I.: A resource elasticity framework for QoS-aware execution of cloud applications. Future Gener. Comput. Syst. **37**(1), 14–25 (2014)
13. Amazon Elastic Map Redude [Online]. http://aws.amazon.com/elasticmapreduce. Accessed 17 Oct 2014
14. Rackspace Public Cloud Pricing [Online]. http://www.rackspace.com/cloud/public-pricing. Accessed 17 Oct 2014
15. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. **11**(1), 10–18 (2009)
16. Kohonen, T.: Self-Organization and Associative Memory, vol. 8. Springer Series in Information Sciences. Springer, Berlin (1989)
17. Amazon Elastic Compute Cloud [Online]. http://aws.amazon.com/ec2/. Accessed 19 Oct 2014
18. Sood, S.K.: Function points-based resource prediction in cloud computing. Concurr. Comput. Pract. Exp. (2014). doi:10.1002/cpe.3296

Rajinder Sandhu obtained his M.E. (Software Engineering) from Thapar University, Patiala in 2013 and B.Tech with Distinction from MMEC, Mullana in 2011. Presently he is pursuing his Ph. D. in cloud computing from Guru Nanak Dev University, Amritsar. His research areas are cloud computing, Virtualization and Software Engineering.

Sandeep K. Sood did his Ph.D. in Computer Science & Engineering from IIT Roorkee, India. He completed his M.Tech, Computer Science & Engineering, from G.J.U, Hisar, India. He is currently working as Associate Dean (A.A. & S.W), Head & Associate Professor, Computer Science & Engineering, G.N.D.U. Regional Campus, Gurdaspur. He has 14 years of teaching and 6 years of research experience. He has more than 50 research publication. His work is published and citied in highly reputed journals such as JNCA, Elsevier and Security and Communication Networks, Wiley. He complete a major research project in cloud computing. His citation number according to Google Scholar is 303 with h-index equal to 10 and i10-index equal to 10. His research areas are Network & Information Security (Password based Authentication Protocol) and cloud computing.