Scheduling in Compute Cloud With Multiple Data Banks Using Divisible Load Paradigm

S. SURESH
HAO HUANG
Nanyang Technological University
Singapore

H. J. KIM
Korea University
Seoul, Korea

The main challenge in a compute cloud system is to design a scheduling strategy for heterogeneous computing resources with shared data banks. The cloud user's job arrives at the Web role, which distributes the load to the worker rules for concurrent processing. The worker role retrieves the respective data from the shared data banks. According to divisible load theory, the scheduling problem is formulated as relevant recursive equations and constraints that are derived from the continuity of processing time due to retrieval from multiple data banks. The scheduling problem in a compute cloud is formulated as a linear programming problem. Finally, we present a satellite image classification problem in a compute cloud as an example to show the adequacy of the proposed solution.

Manuscript received March 23, 2013; revised April 4, 2013, November 21, 2013, April 25, 2014, September 28, 2014; released for publication October 11, 2014.

DOI. No. 10.1109/TAES.2014.130201.

Refereeing of this contribution was handled by T. Robertazzi.

The work of H. J. Kim was supported by the National Research Foundation (NRF) grant (No. 2012015587) and IT Research Center (ITRC) Program, Korea University.

Authors' addresses: S. Suresh, H. Huang, School of Computer Engineering, Nanyang Technological University, Singapore, 50 Nanyang Avenue, Singapore 639798, E-mail: (SSundaram@ntu.edu.sg); H. J. Kim, Multimedia Security Lab, Graduate School of Information Security, Korea University, Seoul 136-713, Korea, E-mail: (khj-@korea.ac.kr).

I. INTRODUCTION

Recent advancements in information technology infrastructures, data storage, and the ubiquity of network resources have led to a new paradigm of computing called cloud computing [1]. The term cloud denotes the interconnected and virtualized infrastructures and resources that are dynamically provisioned and presented as an artificial platform for users to run their applications from anywhere in the world [2]. The computing philosophy has been shifted from the use of a distributed system and server to a cloud of distributed resources. Cloud computing focuses mainly on software as a service, hardware as a service, data as a service, the platform as a service, and autonomy, scalability, and flexibility that provides continuity for large-scale service-oriented applications [3, 4]. For more details on cloud computing, refer to [5, 6].

Many scientific and engineering problems are data driven and are computationally intensive, such as satellite image processing, radar and sensor data processing, and spectrum computation. Thus, some of the previously mentioned applications have to handle huge amounts of data. The objective of these data-driven computations is to minimize the processing time of computing loads. Because the compute cloud [7] provides virtual computational service, these loads can be processed by using a compute cloud environment to reduce the total processing time. Currently, many commercial companies provide compute cloud environment (i.e., Amazon EC2 and Microsoft Windows Azure) that allows the users to scale the capacity based on their resource requirements. The compute cloud environment provides heterogeneous computation platforms and multiple data storage units for the user applications to run. The important challenge in a compute cloud environment is to design a scheduling strategy to handle tasks and to process them in a heterogeneous environment with shared data centers.

Most of the research works on scheduling strategy in a cloud environment are based on service level agreement [2, 8]. Recently, in [9], task scheduling in the cloud environment was solved by using computationally intensive game theory approach with time and cost constraints. Research works are available in the homogeneous cloud environment based on quality of service, without considering communication cost and queuing theory [10–12]. Genetic algorithm-based task scheduling has been implemented for the MapReduce dynamic cloud computing environment [13]. In addition, swarm intelligence-based scheduling strategy to initialize the processing time in compute cloud systems has been proposed in [14]. Both genetic algorithm-based and swarm intelligence-based approaches are search-based approaches and are computationally intensive [13, 14].

Recently, in [15], a large-scale polynomial multiplication was presented in compute cloud by using the divisible load theory (DLT) paradigm, where the virtual computer interface handles the data distribution

and also determines the optimal number of resources, which assumes that the load to process arrives at a virtual computation interface, and the load is distributed to cloud resources. In [16], a multiple source fetching, the load fractions to the child processors employ simultaneous communication for load distribution. Here, a closed-form solution to load fractions using DLT is derived under the assumption that all processors start receiving data from all the sources simultaneously, and the processors have the capability to store the complete data.

In a practical compute cloud environment, the loads and tasks to be processed are available in multiple data centers, rather than in a virtual computer interface. Hence, designing a scheduling strategy in a compute cloud with heterogeneous, dynamic resources and multiple data centers is a challenging problem. Also, one should note that multiple sources may not be available at the time of load distribution, which may influence the load distribution significantly, leading to imbalance in resource utilization. Hence, in this paper, we propose a DLT-based load scheduling problem in a cloud with multiple sources with release time.

In this paper, we design a load distribution strategy for data-driven computational loads on a heterogeneous compute cloud environment, which consists of a virtual resource allocator ("Web role"), a set of virtual computers ("worker role"), and a set of data storage centers ("data bank"). The worker role in the compute cloud is equipped with front-end processors for simultaneous computation and communication (i.e., nonblocking mode of communication) with data banks. We assume that the approach described in [17] is used to synchronize the processing data between the data banks without disturbing the load scheduling process. Note that the data banks are not available at the time of load distribution; hence, it will introduce a delay (referred as release time) in the load scheduling. The objective of the scheduling strategy is to find an optimal fraction of loads assigned to each processor from multiple data centers such that total load processing time is a minimum. The problem of divisible load scheduling is formulated as a linear programming problem.

First, we present a numerical example for load scheduling in a compute cloud and explain the advantage of the proposed formulation. Next, we present simulation results for satellite image classification problem in a virtual cloud in a laboratory environment. For this purpose, we consider a portion of satellite image from high-resolution Landsat Thematic Mapper images [18]. Finally, we highlight the extension of several results by using DLT in a compute cloud environment.

The rest of the paper is constructed as follows: Section II presents a brief review of DLT, and Section III describes the compute cloud platform model and definitions. The detailed divisible load scheduling strategy for a compute cloud is given in Section IV. Section V presents performance evaluation and experimental study in a virtual cloud platform for a satellite image processing problem. Finally, Section VI concludes the work.

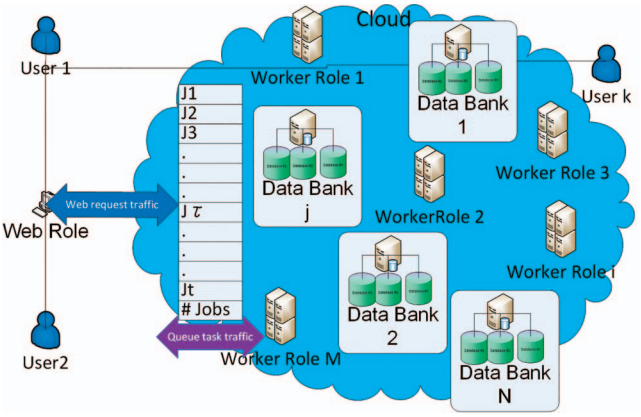

Fig. 1. Compute cloud environment.

II. BRIEF REVIEW OF DLT

The data-driven computing application in the distributed computing system can be partitioned into smaller frames and can be processed simultaneously without any precedence. DLT helps to schedule computationally intensive processing loads with communication delays [19]. The scheduling problems can be addressed in a linear chain [20], single-level tree [21], and bus network [22]. In addition, many research articles were published in the literature with start-up delay [23, 24], buffer constraints [15, 25, 26, 49], release time [27, 28], arbitrary location [29], equal allocation [30], nonlinear loads [31–33], distributed processing [34], various communication loads [35–38], and practical applications [39, 40]. In [41–43], an adaptive divisible load model with iterative approach is used to solve scheduling optimal workload in grid applications. Ten reasons to use a divisible load paradigm have been presented in [44]. A complete review on the development of DLT can be found in [45, 46].

III. MATHEMATICAL MODELLING OF A COMPUTE CLOUD PLATFORM

A compute cloud environment, shown in Fig. 1, consists of heterogeneous or homogeneous virtual computer interfaces (worker roles) and multiple data storage centers (data bank). For better utilization of a compute cloud platform, the user submits the request for load scheduling at a virtual resource allocator (Web role). Resource allocations assume that the data-driven computational load is stored in multiple data centers and the data can be synchronized effectively without disturbing the load distribution process [17]. Note that the number of data banks is much smaller than the number of worker roles available for computation. Once a request is received from a user or application, the virtual resource allocator divides the computational load and assigns the load fractions to worker roles. Each worker role retrieves the load fractions from the available data banks in multiple

TABLE I
Notations and Descriptions

A_i	Inverse processing speed to worker role W_i
G_j	Inverse transmitting speed to the data bank D_j, through a wide-area network
J	Total processing load
N	Number of data banks in one job scheduling round
M	Number of worker roles in one job scheduling round
R_j	Release time of data bank D_j
T_i	Data processing time for ith worker role (W_i)
T_O	Total load processing time
β_{ij}	The fraction of load retrieved from the data bank (D_j) by a worker role (W_i)
μ_i	The total fraction of the processing load assigned to a worker role (W_i) ($i = 1, 2, \ldots, M$)

installments such that the processing time is a minimum. The installment size depends on mode of communication, communication link speed, release time of the data banks, and computing power of worker role. The worker role in the compute cloud is equipped with front-end processors for simultaneous computation and communication (i.e., nonblocking mode of communication) with data banks.

In our analysis, the compute cloud system is considered to have M heterogeneous computational worker roles ($W_1, W_2, \ldots, W_M$) and N data banks ($D_1, D_2, \ldots, D_N$). Note that $N \ll M$ in actual cases. The worker role accesses the data banks to retrieve their respective load fractions one-by-one. Note that the data banks may not be available at the beginning of load distribution. The delay in data bank availability for load retrieval is called release time. In our formulation, we assume that the Web role assigns the sequence of load distribution as ($W_1, W_2, \ldots, W_M$), and each worker role retrieves their portion of loads from data banks in the order of ($D_1, D_2, \ldots, D_N$). The release time of data bank (R_j) and communication speed parameter (G_j) influence the size of load fractions retrieved from the data bank. The Web role partitions the total computational load (J) into M fractions ($\mu_1, \mu_2, \ldots, \mu_M$) and assigns them to the worker roles ($W_i; i = 1 \ldots M$). The worker role (W_i) retrieves its own fraction (μ_i) from N data banks ($\beta_{i1}, \beta_{i2}, \ldots, \beta_{iN}$).

Load distribution is defined as an M tuple of total processing load ($\mu_1, \mu_2, \ldots, \mu_M$) assigned to the worker roles ($W_1, W_2, \ldots, W_M$) in the compute cloud. The total processing load is given by

$$J = \sum_{i=1}^{M} \mu_i = \sum_{i=1}^{M} \sum_{j=1}^{N} \beta_{ij} \quad i = 1, \ldots, M,\ j = 1, 2, \cdots, N \tag{1}$$

Finish time of the worker role (T_i) is the time difference between the time instance at which the worker role (W_i) stops computing and the time instance at which the worker role initiates the task.

Processing time (T_o) is defined as the total time taken the Web role to compute the entire processing load in a compute cloud system, i.e., $T_o = max\{T_1, T_2, \ldots, T_M\}$.

The notations and descriptions used in the paper are given in Table I.

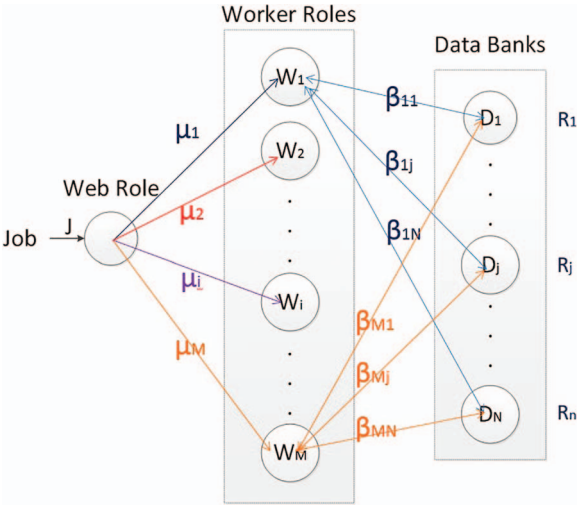

Fig. 2. Compute cloud environment with M worker roles and N data banks.

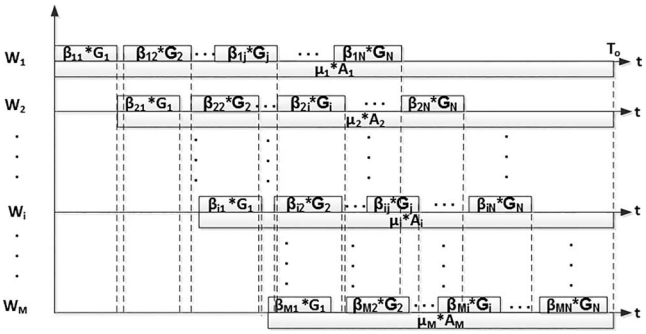

Fig. 3. Timing diagram describing load scheduling process in compute cloud environment.

IV. LOAD DISTRIBUTION SCHEDULING STRATEGY

The scheduling problem in a compute cloud environment can be captured in Fig. 2. The Web role divides the computing load into smaller fractions and assigns the tasks to the worker roles. The worker role retrieves the data from data banks and completes the computation process. Now, we formulate the scheduling problem in the compute cloud environment.

The load distribution processes by the Web role and retrieval sequence of load fractions by the worker role are illustrated by means of the timing diagram, as shown in Fig. 3.

Let T_o describe the optimized total load processing time. For a minimal computational time, all worker roles stop computing at the same time, as stated in divisible load scheduling theory [19].

Hence, from the timing diagram, we can write the following equation:

$$T_o - R_1 - \sum_{j=1}^{i-1} \beta_{j1} \cdot G_1 - \sum_{j=1}^{N} \beta_{ij} \cdot A_i \geq 0; i = 1, \ldots, M \tag{2}$$

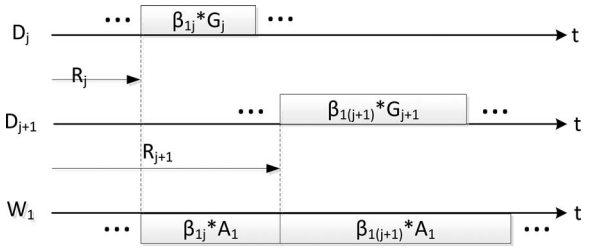

Fig. 4. Timing diagram of worker role processing by accessing every adjacent data bank pair.

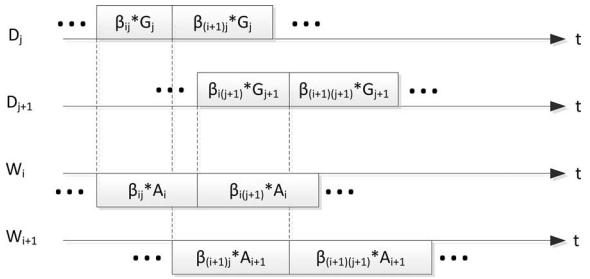

Fig. 5. Timing diagram of adjacent processing tasks.

A. Constraints Imposed by Release Time (R_j)

Note that the first worker role retrieves its fractions of load from data banks. The release time of the data bank will affect the retrieval of the load by the first worker role. Various factors, such as storage, maintenance, consistency, retrieval by other hosts, and fault recovery by overheating, may affect the availability of a data bank for load retrieval.

If the release time R_{j+1} of the data bank D_{j+1} is higher than the computation of load β_{ij} by the worker role W_1, then the worker role will be idle for some time. In addition to the release time of the data banks, the size of load retrieval from adjacent data banks (D_j, D_{j+1}) to adjacent worker roles (W_i, W_{i+1}) may result in idle time. To minimize the idle time or maximize the resource utility, the data bank D_j allocates work load to the worker role (β_{ij}). From the timing diagram shown in Fig. 4, we can see that the difference in release times of two adjacent data banks (D_j, D_{j+1}) influences the component load retrieval by the worker role for computation, i.e.,

$$\beta_{1j} \cdot A_1 \geq R_{j+1} - R_j \quad j = 1, 2, \cdots, N-1 \tag{3}$$

The timing diagram, describing the continuity constraint on the worker role, is shown in Fig. 5. From this figure, we can derive the following equation

$$\beta_{ij} \cdot A_i + \beta_{i(j+1)} \cdot G_{j+1} \leq \beta_{ij} \cdot G_j + \beta_{(i+1)j} \cdot A_{i+1}$$
$$i = 1, \ldots, M-1, \; j = 1, 2, \cdots, N-1 \tag{4}$$

This equation applies for two adjacent worker roles and two adjacent data banks and results in $(M-1) \cdot (N-1)$ number of constraint equations. Note that because obtaining a closed-form expression for load fractions (β_{ij}) in the divisible load scheduling problem is difficult, we propose this load scheduling problem with multiple data banks as an optimization problem. The optimization problem is defined as the following:

Given the number of worker roles (M), number of data banks (N), and each of load retrievals ($W_1, W_2, \cdots, W_M$) from data banks ($D_1, D_2, \cdots, D_N$), find the load fractions assigned to each worker role from each data bank such that the total load processing time is minimized.

Minimize T_o such that

$$T_o - R_1 - \sum_{j=1}^{i-1} \beta_{j1} \cdot G_1 - \sum_{j=1}^{N} \beta_{ij} \cdot A_i \geq 0 \quad i = 1, 2, \ldots, M$$

$$\beta_{1j} \cdot A_1 \geq R_{j+1} - R_j \quad j = 1, 2, \cdots, N-1$$

$$J = \sum_{i=1}^{M} \mu_i = \sum_{i=1}^{M} \sum_{j=1}^{N} \beta_{ij} \quad i = 1, \ldots, M, \; j = 1, 2, \cdots, N$$

$$\beta_{ij} \cdot A_i + \beta_{i(j+1)} \cdot G_{j+1} \leq \beta_{ij} \cdot G_j + \beta_{(i+1)j} \cdot A_{i+1}$$

$$i = 1, \ldots, M-1, \; j = 1, 2, \cdots, N-1$$

This linear programming problem has $(M \times N + 1)$ variables and $(M \times N + 1)$ constraints. The variables in our problem are the processing time (T_o) and the load fractions $\{(\beta_{11}, \beta_{12}, \cdots \beta_{1N}), \cdots, (\beta_{M1}, \beta_{M2}, \cdots \beta_{MN})\}$. The solution of the previously mentioned problem is a point in $(M \times N + 1)$ dimensional space.

We provide a numerical example for a better understanding of the proposed scheduling strategy for a compute cloud environment by using a linear programming formulation.

1) *Numerical Example 1*: Consider a compute cloud environment with three worker roles and two data banks. The computing speed parameters of the worker roles are $A_1 = 4$, $A_2 = 5$, and $A_3 = 6$. The communication speed parameters of the data banks are $G_1 = 0.2$ and $G_2 = 0.4$. The release times of the data banks are $R_1 = 10$ and $R_2 = 50$. Without loss of generality, we assume that the entire processing load ($J = 100$) is stored in data banks before the start of the computation process.

All worker roles that stop computing at the same time impose three inequality constraints, and the release time of data banks imposes one inequality constraint. Total load processing and continuity in worker role computation imposes three equality constraints. The objective is to find the load fractions assigned to each worker role from data banks such that the total load processing time is a minimum. Hence, the load scheduling problem in a compute cloud can be formulated as a linear programming problem as given in the following:

Minimize T_o such that

$$T_o - R_1 - (\beta_{11} + \beta_{12}) \cdot A_1 \geq 0$$

$$T_o - R_1 - \beta_{11} \cdot G_1 - (\beta_{21} + \beta_{22}) \cdot A_2 \geq 0$$

$$T_o - R_1 - (\beta_{11} + \beta_{21}) \cdot G_1 - (\beta_{31} + \beta_{32}) \cdot A_3 \geq 0$$

$$\beta_{11} \cdot A_1 \geq R_2 - R_1$$

TABLE II
Processing Time and Total Load Fractions Assigned to Worker Roles from Different Data Banks

T_o	β_{11}	β_{12}	β_{21}	β_{22}	β_{31}	β_{32}
173.9	10	30.974	10.078	22.301	9.5491	17.098

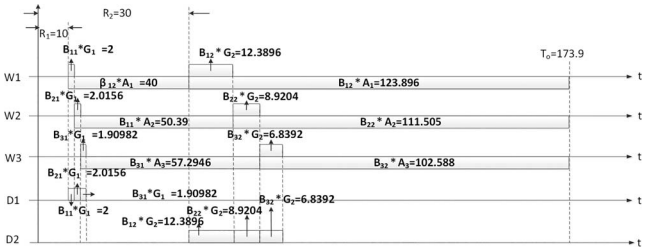

Fig. 6. Timing diagram illustrating results of numerical example.

$$\beta_{11} + \beta_{12} + \beta_{21} + \beta_{22} + \beta_{31} + \beta_{32} = J$$

$$\beta_{11} \cdot A_1 + \beta_{12} \cdot G_2 \leq \beta_{11} \cdot G_1 + \beta_{21} \cdot A_2$$

$$\beta_{21} \cdot A_2 + \beta_{22} \cdot G_2 \leq \beta_{21} \cdot G_1 + \beta_{31} \cdot A_3$$

Based on the previous formulation, load fractions assigned to each worker role from each data bank and total load processing time are given in Table II. For a better understanding of the results, we illustrate the process of load retrieval from each data bank to three worker roles and their computation time in the timing diagram in Fig. 6. From the figure, we can see that the worker role computation is continuous. and all worker roles stop computing at the same time.

It has been proven in the DLT literature that the sequence of distribution influences the processing time significantly and has been solved for load scheduling in a single installment for tree, bus, and linear network [19]. The presence of multiple fractions of load retrieval from different data banks and release time of data banks in a cloud environment complicates the optimal sequencing problem. To study the effect of sequence of load retrieval from data banks, we have conducted experiments in which the worker roles start retrieving the load from data bank 2 and then retrieve the load fraction from data bank 1. The order of load distribution remains the same. Note that the release time of the data banks is modified as $R_1 = 50$ and $R_2 = 10$. The total load processing time increases to 175.45. Note that the release time of data banks also influences the sequence of load retrieval. One should use the data banks with smaller release time first, and if two data banks have similar release time, then the one with faster communication speed is used first for the load retrieval. The problem of finding optimal sequence of load retrieval in the presence of multiple data banks with release time is NP-hard, and in [47], the mentioned multi-installment DLT scheduling is also NP-hard.

2) *Numerical Example 2*: To study the effect of release time, we consider a compute cloud environment with three homogeneous worker roles and two data banks. The computing speed parameters of the worker roles are $A_1 = 1.1$, $A_2 = 1.1$, and $A_3 = 1.1$. The communication speed parameters of the data banks are $G_1 = 0.2$ and $G_2 = 0.4$. The release times of the data banks are $R_1 = 10$ and $R_2 = 10$. For this problem also, we assume that the entire processing load ($J = 100$) is stored in data banks before the start of the computation process. Note that the release time constraint given in (3) reduces to $\beta_{11} \geq 0$. Based on our formulation, the load fractions assigned to each worker role from each data bank and total load processing time are $\beta_{11} = 0$, $\beta_{12} = 19.8198$, $\beta_{21} = 19.8198$, $\beta_{22} = 36.3364$, $\beta_{31} = 16.5165$, and $\beta_{32} = 7.5075$. The total load processing time is 43.97. From the results, we can see that the first worker role does not receive any load fraction from the first data bank.

V. PERFORMANCE EVALUATION AND DISCUSSION

In our simulation, we consider a compute cloud with 10 worker roles and five different data banks for the satellite image processing application. We generate random values of worker role inverse processing speed, the inverse transmitting speed of data banks, and release time of data banks such that the computation time is shorter than the communication time. These parameters are given in Table III. The computational load arrives at the Web role, and the Web role identifies the worker roles and the data banks for load scheduling. The worker roles receive data from each data bank and stops computing at the same time.

The performance of the compute cloud with different data banks is evaluated for a heterogeneous and homogeneous worker role. For all our studies, we use total load processing time as a performance metric. First, we present the performance of the heterogeneous worker role in the compute cloud environment.

Fig. 7 shows the processing time versus the number of worker roles for different number of data banks in a compute cloud environment. From the figure, we can see that the processing time decreases with an increase in the number of worker roles for a given data bank. Beyond a certain number of worker roles, total load processing time does not decrease further. From Fig. 7, we can also see that the processing time decreases significantly when we increase the number of data banks with a fixed number of worker roles. For example, three worker roles with two data banks require 203.13 time units for processing the total load, whereas three worker roles with three data banks require only 186.79 time units. We can also observe from the figure that after a certain number of data banks, the processing time does not reduce significantly.

Now, we present the processing time for different number of homogeneous worker roles and heterogeneous data banks. For our simulation study, we have used worker roles with computation speed parameter as 1.1 and the rest of the parameters, as given in Table III. The total load processing time versus number of worker roles for different number of data banks is shown in Fig. 8. From

TABLE III
Cloud System Simulation Parameters

Data Bank		Worker Role
Release Time (R_i)	Communication Speed (G_i)	Computation Speed (A_i)
(4,5,6,7,8)	(0.5,0.6,0.7,0.8,0.9)	(1,1.1,1.2,1.3,1.4, 1.5,1.6,1.7,1.8,1.9)

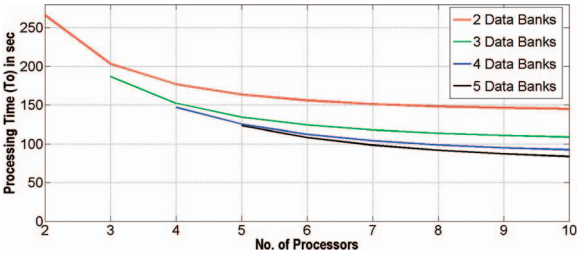

Fig. 7. Processing time versus number of heterogeneous worker roles in compute cloud environment.

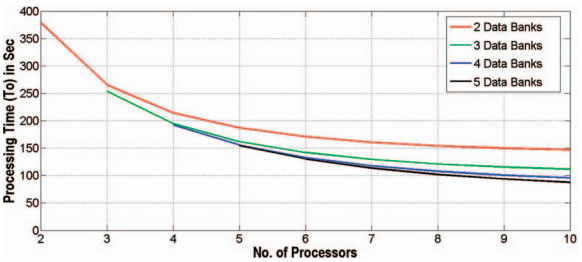

Fig. 8. Processing time versus number of homogeneous worker roles in compute cloud environment.

the figure, we can observe that total processing time decreases with the increase in the number of worker roles and data banks as in heterogeneous case. The reduction in total load processing time becomes smaller beyond a certain number of worker roles and data banks.

Further, we conducted experiments by varying the communication speed parameter (G) of the homogeneous data banks as 0.5, 0.6, 0.7, and 0.8. The rest of the parameters are kept constant as in the earlier homogeneous scenario, and number of data banks is kept at two. The processing time for different number of worker roles for different communication speed parameters are shown in Fig. 9. From the figure, we can see that the reduction in processing time decreases with the increase in the data bank communication speed parameter. Also, the rate of decrease in processing time to increase in number of processors decreases with an increase in the communication speed parameter.

To study the effect of total processing load (J), we conduct a study by varying the processing load in a homogeneous environment with two data banks. The parameters are set as G = 0.5, $A = 1.1$, $R_1 = 3$, and $R_2 = 4$, respectively. The processing time variation for a various number of processors with different total processing load is shown in Fig. 10. From the figure, we

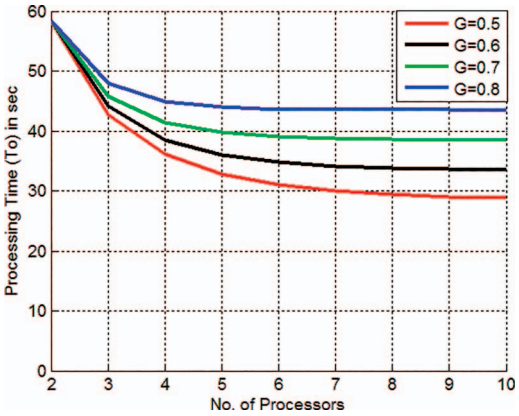

Fig. 9. Processing time versus number of homogeneous worker roles for different communication speed parameters.

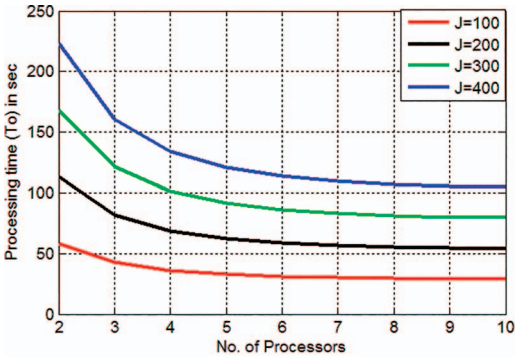

Fig. 10. Processing time versus number of homogeneous worker roles for different processing load sizes.

can see that increasing the total processing load reduces the rate of decrease in processing time.

Now, we conduct a study on computational complexity of the proposed approach using MATLAB. For this purpose, we consider a homogeneous compute cloud environment with communication speed parameter (G) for all links is 0.1, and a computation speed parameter (A) for all worker roles is 1.1. The number of worker roles (M) varies between 100 and 500, and the number of data banks (N) is kept as 20 and 30, respectively. The release time of the data banks is initialized appropriately (in steps of 10). The time taken to compute the respective load fraction from each data bank to each processor is plotted against the number of worker roles in Fig. 11. From the figure, we can see that the time increases with an increase in the number of worker roles and data banks, but the execution time is of the order of few seconds.

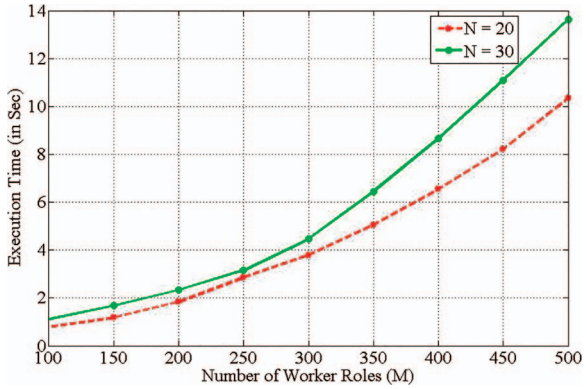

Fig. 11. Execution time versus number of homogeneous worker roles in compute cloud environment.

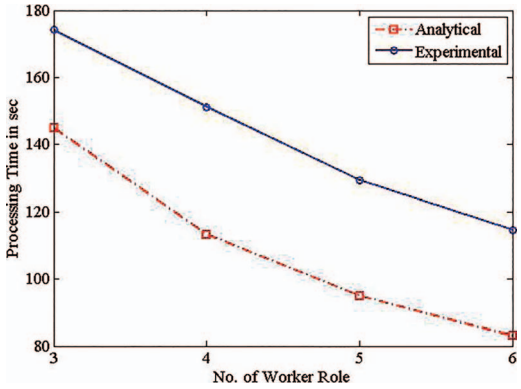

Fig. 12. Analytical and experimental processing time for satellite image processing problem in virtual cloud environment.

A. Satellite Image Processing Application

For the experimental study, we developed a virtual cloud in a laboratory environment with two servers acting as a data bank and six work stations acting as a worker role for data processing. The HP servers and workstations are running in an HPUX 11v3 operational environment connected through 10 Gbps Ethernet link. For experimental study, we consider a satellite image classification problem [18]. A high-resolution multispectral Landsat 7 Thematic Mapper image portion covering 50×50.75 km^2 is considered for the study. The spatial resolution of images is 30 m, and each pixel contains seven spectral frequencies and is used as an input feature to online learning neural classifiers. The worker roles contain classifier model developed by using an online sequential learning classifier [48] and are expected to estimate the class label for a given pixel in a portion of the satellite image. For more details about the data set, one should refer to [18].

The communication parameters are estimated by averaging the time taken to send a 1 000 000 pixels (i.e., 7 000 000 floating point values) between the data banks and workstations for 100 times. Similarly, the computation parameter is estimated by averaging the computational time taken by the workstations to process the unit load for 100 times. The communication (G_j) and computation (A_i) speed parameters are 0.7 and 4.9, respectively. The release time for data banks (D_1 and D_2) are assumed to be 10 and 20 s, respectively. The total number of pixels in the image region are 84 583 000, and one unit is 10^6. These parameters are used in our numerical example and experimental study. The total pixels are divided into smaller fractions as obtained from our formulation, and the load fractions are rounded to nearest integer. The workstations retrieve these pixels from data banks and classify them into respective classes. The experimental processing and analytical time taken to classify the satellite image with two data banks and three worker roles are 174.18 and 144.85 s, respectively. The load fractions assigned to the worker from each data bank are {2.1277, 5.7672, 8.2753, 26.564, 22.608, 19.241}. The difference between the experimental and analytical times is due to overheads in communication, data reformatting, synchronization, and so on.

The analytical and experimental time taken to process the satellite image for different number of worker roles is shown in Fig. 12. From the figure, we can see that the processing time decreases with an increase in the number of worker roles. As mentioned earlier, the analytical time is smaller than the experimental time due to communication and computation components included in the formulation.

VI. CONCLUSION

In this paper, we have presented a scheduling algorithm to compute a cloud environment with multiple data banks. First, the problem of load scheduling in a compute cloud is formulated in a divisible load scheduling framework. Next, we converted the recursive equations and continuity of processing into constraints and formulated the scheduling problem as a linear programming problem. A linear programming technique has two advantages: one is to make the best possible use of available productive resources, and the other is to solve complex problems by breaking them into solvable parts. Finally, we have presented a numerical satellite image processing example, and the experimental simulation results highlight the advantage of the proposed solution. Also, we have conducted a simulation study to understand the computational complexity of the proposed approach, and the results clearly highlight that it is of the order of seconds. Future direction will explore the possibilities of handling multiple job scheduling with different priorities.

REFERENCES

[1] Buyya, R., Yeo, C. S., Venugopal, S., Breberg, J., and Brandic, I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computing Systems*, **25**, 6 (2009), 599–616.

[2] Daniel, D., and Lovesum, S. P. J. A novel approach for scheduling service request in cloud with trust monitor.

In *Proceedings of the International Conference on Signal Processing, Communication, Computing and Networking Technologies*, Thuckalay, India, July 21–22, 2011, 509–513.

[3] Wang, X., Wang, B., and Huang, J.
Cloud computing and its key techniques.
In *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering*, Shanghai, China, June, 10–12, 2011, 404–410.

[4] Yang, Y., Choi, J. Y., Choi, K. M., Gannon, P. M., and Kim, D. S.
BioVLAB-microarray: microarray data analysis in virtual environment.
In *Proceedings of the IEEE Fourth International Conference on eScience*, Dec. 7–12, 2008, 159–165.

[5] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M.
Above the clouds: a Berkeley view of cloud computing.
University of California, Berkeley, Tech. Rep. No. UCB/EECS-2009028, 2009.

[6] Mell, P., and Grance, T.
The NIST definition of cloud computing.
NIST Special Publication 800-145. NIST, US Department of Commerce, 2011.

[7] Grossman, R. L., Yunhong, G., Sabala, M., and Wanzhi, Z.
Compute and storage clouds using wide area high performance networks.
Future Generation Computer Systems, **25**, 2 (2009), 179–183.

[8] Li, Q., and Guo, Y. K.
Optimization of resource scheduling in cloud computing.
In *Proceedings of the International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Timioara, Romania, Sep. 23–26, 2010, 315–320.

[9] Wei, G. Y., Vasilakos, A. V., and Xiong, N. X.
Scheduling parallel cloud computing services: an evolutional game.
In *Proceedings of the International Conference on Information Science and Engineering*, Nanjing, China, Dec. 26–28, 2009, 376–379.

[10] Huang, Q. Y., and Huang, T. L.
An optimistic job scheduling strategy based on QoS for cloud computing.
In *Proceedings of the International Conference on, Intelligent Computing and Integrated Systems*, Guilin, China, Oct. 22–24, 2010, 673–675.

[11] Murata, Y., Egawa, R., Higashida, M., and Kobayashi, H.
A history-based job scheduling mechanism for the vector computing cloud.
In *Proceedings of the IEEE/IPSJ International Symposium on Applications and the Internet*, Seoul, Korea, July 19–23, 2010, 125–128.

[12] Hu, J. H., Gu, J. H., Sun, G. F., and Zhao, T. H.
A scheduling strategy on load balancing of virtual machine resources in a cloud computing environment.
In *Proceedings of the Third International Symposium on Parallel Architectures, Algorithms and Programming*, Dalian, China, Dec. 18–20, 2010, 89–96.

[13] Ge, Y. G., and Wei, G. Y.
GA-based task scheduler for the cloud computing systems.
In *Proceedings of International Conference on Web Information Systems and Mining*, Sanya, China, Oct. 23–24, 2010, 181–186.

[14] Pandey, S., Wu, L. L., Guru, S. M., and Buyya, R.
A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments.
In *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications*, Perth, WA, April 20–23, 2010, 400–407.

[15] Ganesh, N. I., Bharadwaj, V., and Sakthi, G. K.
On handling large-scale polynomial multiplications in compute cloud environments using the divisible load paradigm.
IEEE Transactions on Aerospace and Electronics Systems, **48**, 1 (2012), 820–831.

[16] Wong, H. M., Yu, D., Veeravalli, B., and Robertazzi, T. G.
Data intensive grid scheduling: multiple sources with capacity constraints.
In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, Marina del Ray, CA, 2003.

[17] El-Gorashi, T. E. H., and Elmirghani, J. M. H.
Distributed storage scenario in a wide area WDM mesh architecture under heterogeneous traffic.
In *Proceedings of the International Conference on Optical Network Design and Modeling*, Braunschweig, Germany, Feb. 18–20, 2009, 1–6.

[18] Suresh, S., Mani, V., Omkar, S. N., and Sundararajan, N.
Multi-spectral satellite image classification using an evolving neural network approach.
Journal Aerospace Technology, **58**, 4 (2006), 287–304.

[19] Veeravalli, B., Debasish, G.,Venkataraman, M., and Robrtazzi, T. G.
Scheduling Divisible Loads in Parallel and Distributed Systems. Los Alamitos, CA: IEEE Computer Society Press, 1996.

[20] Cheng, Y. C., and Robertazzi, T. G.
Distributed computation with communication delay distributed intelligent sensor networks.
IEEE Transactions on Aerospace and Electronic Systems, **24**, 6 (1988), 700–712.

[21] Cheng, Y. C., and Robertazzi T. G.
Distributed computation for a tree network with communication delays.
IEEE Transactions on Aerospace and Electronic Systems, **26**, 3 (1990), 511–516.

[22] Bataineh, S., and Robertazzi, T. G.
Bus-oriented load sharing for a network of sensor driven processors.
IEEE Transactions on Systems, Man and Cybernetics, **21**, 5 (1991), 1202–1205.

[23] Suresh, S., Mani, V., and Omkar, S. N.
The effect of start-up delays in scheduling divisible loads on bus networks: an alternate approach.
Computers Mathematics with Applications, **40**, 10 (2003), 1545–1557.

[24] Bharadwaj, V., Li, X., and Ko, C. C.
On the influence of start-up costs in scheduling divisible loads on bus networks.
IEEE Transactions on Parallel and Distributed Systems, **11**, 12 (2000), 1288–1305.

[25] Li, X. L., Bharadwaj, V., and Ko, C. C.
Divisible load scheduling on a hypercube cluster with finite-size buffers and granularity constraints.
In *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*, Brisbane, Australia, 2001, 660–667.

[26] Suresh, S., Mani, V., Omkar, S. N., and Kim, H. J.
Divisible load scheduling in distributed system with buffer constraints: genetic algorithm and linear programming approach.
The International Journal of Parallel, Emergent and Distributed Systems, **21**, 5 (2006), 303–321.

[27] Suresh, S., Mani, V., Omkar, S. N., and Kim, H. J.
A real coded genetic algorithm for data partitioning and scheduling in networks with arbitrary processor release time.

Advances in Computer Systems Architecture, **3740** (2005), 529–539.

[28] Bharadwaj, V., Li, H. F., and Radhakrishnan, T.
Scheduling divisible loads in bus network with arbitrary release time.
Computers and Mathematics with Applications, **32**, 7 (1996), 57–77.

[29] Kyong, Y. T., and Robertazzi, T. G.
Greedy signature processing with arbitrary location distributions: a divisible load framework.
IEEE Transactions on Aerospace and Electronic Systems, **48**, 4 (2012), 3027–3041.

[30] Ko, K., and Robertazzi, T. G.
Equal allocation scheduling for data intensive applications.
IEEE Transactions on Aerospace and Electronic Systems, **40**, 2 (2004), 695–705.

[31] Suresh, S., Run, C., Kim, H. J., Robertazzi, T. G., and Kim, Y. I.
Scheduling second order computational load in master-slave paradigm.
IEEE Transactions on Aerospace and Electronics Systems, **48**, 1 (2012), 780–793.

[32] Hung, J. T., and Robertazzi, T. G.
Distributed scheduling of nonlinear computational loads.
In *Proceedings of the Conference on Information Sciences and Systems*, Baltimore, MD, Mar. 2003.

[33] Suresh, S., Kim, H. J., Run, C., and Robertazzi, T. G.
Scheduling nonlinear divisible loads in a single level tree network.
The Journal of Supercomputing, **61**, 3 (2012), 1068–1088.

[34] Fard, F. N., Broumandnia, A., and Mohammadi, S.
Distributed image processing scheduling in heterogeneous computing network systems.
In *Proceedings of the 13th International Workshop on Mechatronics*, Paris, France, 2012, 1–5.

[35] Kim, H, J., and Mani, V.
Divisible load scheduling in single-level tree network in non-blocking mode of communication.
Computer & Mathematics with Applications, **46**, 10–11 (2003), 1611–1623.

[36] Suresh, S., Mani, V., Omkar, S. N., and Kim, H. J.
An equivalent network for divisible load scheduling in nonblocking mode of communication.
Computer and Mathematics with Applications, **49**, 9–10 (2005), 1413–1431.

[37] Hung, J. T., and Robertazzi, T. G.
Switching in sequential tree networks.
IEEE Transactions on Aerospace and Electronic Systems, **40**, 3 (2004), 968–982.

[38] Suresh, S., Mani, V., Omkar, S. N., Kim, H. J., and Sundararajan, N.
New load distribution strategy for linear network with communication delays.
Mathematics and Computers in Simulation, **79**, 5 (2009), 1488–1501.

[39] Suresh, S., Mani, V., Omkar, S. N., and Kim, H. J.
Parallel video processing using divisible load scheduling.
Journal of Broadcast Engineering, **10**, 1 (2005), 83–102.

[40] Suresh, S., Omkar, S. N., and Mani, V.
Parallel implementation of back-propagation algorithm in network of workstations.
IEEE Transactions on Parallel Distributed Systems, **16**, 1 (2005), 23–34.

[41] Othman, M., Abdullah, M., Ibrahim, H., and Subramaniam, S.
Adaptive divisible load model for scheduling data-intensive grid applications.
Computational Science, **4487** (2007), 446–453.

[42] Othman, M., Abdullah, M., Ibrahim, H., and Subramaniam, S.
A2DLT: divisible load balancing model for scheduling communication-intensive grid applications.
Lecture Notes in Computer Science, **5101** (2008), 246–253.

[43] Abdullah, M., Othman, M., Ibrahim, H., and Subramaniam, S.
Optimal workload allocation model for scheduling divisible data grid applications.
Future Generation Computer Systems, **26**, 7 (July 2010), 971–978.

[44] Robertazzi, T. G.
Ten reasons to use divisible load theory.
Computer, **36**, 5 (2003), 63–68.

[45] Ghose, D., and Robertazzi, T.
Foreword (Special issue of *Cluster Computing* on divisible load scheduling).
Cluster Computing, **6**, 1 (2003), 5.

[46] Ghanbari, S., and Othman, M.
(2014). Comprehensive review on divisible load theory: concepts, strategies, and approaches.
Mathematical Problems in Engineering. [Online]. 2014 Available: http://www.hindawi.com/journals/mpe/2014/460354/.

[47] Yang, Y., Casanova, H., Drozdowski, M., Lawenda, M., and Arnaud, A.
On the complexity of multi-round divisible load scheduling.
Institut national de recherche en informatique et en automatique. Rep. 6096, 2007.

[48] Babu, G. S., and Suresh, S.
Sequential projection-based metacognitive learning in a radial basis function network for classification problems.
IEEE Transactions on Neural Networks and Learning Systems, **24**, 2 (2013), 194–206.

[49] Li, X., Bharadwaj, V., and Ko, C. C.
Divisible load scheduling on single-level tree networks with buffer constraints.
IEEE Transactions on Aerospace and Electronic Systems, **36**, 4 (2000), 1298–1308.

Sundaram Suresh (M'08—SM'10) received a B.E. degree in electrical and electronics engineering from Bharathiyar University in 1999 and M.E. (2001) and Ph.D. (2005) degrees in aerospace engineering from the Indian Institute of Science, India. He was a postdoctoral researcher in the School of Electrical Engineering, Nanyang Technological University from 2005 to 2007. From 2007–2008, he was in Institut national de recherche en informatique et en automatique, Sophia Antipolis, France, as an European Research Consortium for Informatics and Mathematics research fellow. He was at the Korea University for a short period as a visiting faculty in industrial engineering. From January 2009 to December 2009, he was at the Indian Institute of Technology, Delhi, as an assistant professor in the Department of Electrical Engineering. Currently, he is working as an assistant professor at the School of Computer Engineering, Nanyang Technological University, Singapore, since 2010. His research interest includes cognitive computing, flight control, unmanned aerial vehicle design, machine learning, applied game theory, optimization, and computer vision.

Hao Huang (M'13) received a B.E. degree in electrical engineering and automation from Hebei University of Technology in 2002, an M.Sc. degree in computing and information networks from the University of Essex 2005, and is now studying at Nanyang Technological University as a part-time Ph.D. student. Currently, he is working in Hewlett Packard Singapore Ltd. as a specialist level engineer.

Hyoung Joong Kim (M'80) received his B.S., M.S., and Ph.D. degrees from Seoul National University, Seoul, Korea, in 1978, 1986, 1989, respectively. He joined the faculty of the Department of Control and Instrumentation Engineering, Kangwon National University, Korea, in 1989. Since 2006, he has been a professor at the Graduate School of Information Security, Korea University, Korea. His research interests include parallel and distributed computing, multimedia computing, multimedia security, and electronic warfare systems.