Study and Analysis of Cloud Aided Remote Sensing Multiprocessing System (CARMS) with Task Allocation

M. Zubair khan, Geeta Devi, and Yasser M. Alginahi

Abstract—**Cloud computing is one of the promising and successful technology in this technological era and because of the limitation of remote sensing the concept called Cloud Aided Remote Sensing Multiprocessing System (CARMS) came into existence. It is the combination of technologies called Remote sensing and Cloud Computing, which makes Internet of everything enabler possible (ubiquitous computing). In CARMS scenario, system can have different types of tasks or requests and some maybe requested at the same instance of time. In such cases, it is important that the system should serve maximum possible tasks gaining profit to the sensing services at clouds and providing user satisfaction at the same time. Hence, for the sensing services of sensors in cloud, an optimal scheduling or task allocation scheme has to be developed so that multiple requests or tasks may get response and therefore, these tasks can be scheduled, processed and handled properly without any delay. Thus, the proposed work provides an algorithm that allows efficient task allocation which aims at providing efficient distribution of tasks to the Virtual Machines in the cloud.**

Index Terms—**Remote sensing (RS), cloud computing (CC), internet of everything enabler (IOE), cloud aided remote sensing multiprocessing system (CARMS), distributive sensor cloud system (DSCS).**

I. Introduction

The Major role or purpose of Remote Sensing is to acquire or collect data remotely or in real time from everywhere without needing physical field visits and this is done with the help of sensory things or objects (sensors, SAN etc). As sensors play a very important role in providing a very useful information so they are being utilized in different applications like defense for military border tracking and surveillance, weather condition, healthcare, Natural disaster relief etc. However, it have to suffer from or have many issues or limitations like short communication range, security and privacy, power consideration, storage capacity, processing capabilities etc. As the Cloud Computing is the emerged ongoing research technology which allow efficient computation by centralizing storage and memory. Most of the organizations are using this technology as it is easy and economical to use. Currently, many researchers are working on checking the suitability of this technology and its implemention for avoiding the limitation of Remote Sensing also. The new technology CARMS emerged from the combination of cloud computing and remote sensing. Therefore, CARMS can be viewed as an emerging technology that has great potential for enabling IOE (*Internet* of Everything) therefore enabling smart cloud services. And this technology is then provided to all the devices such as sensors of mobile devices, smart phones, portable terminal and so on, many of the applications have been developed which are using CARMS like Nimbits [1], Pachube Platform [2], Thing Speak [3]-[15]. It will give rise to better remote sensing applications to the society, once it is successfully developed without any flaws. So, this paper presents an overview of CARS with the proposed task allocation algorithm.

The rest of this paper is organized as follows: Section II, gives introduction to CARMS; Section III discusses CARMS existing architecture; Section IV refers to the CARMS Services Models; Section V explains the proposed efficient task allocation algorithm and Section VI presents the performance appraisal of the proposed algorithm. Finally, Section VII concludes the paper.

II. Background

CARMS is the combination of cloud computing and remote sensing, which provide cloud aided remote sensing services to users through the *Internet* and Sensor or Sensing devices. Remote Sensing is "a science of Acquiring, Processing and Interpreting Images and related data that are obtained from ground based, air-or space instruments [1], [16]." Further, Cloud Computing is "*Internet*-based computing, whereby shared resources, software and information are provided to computers and other devices on-demand [2]." As shown in Fig. 1, CARMS can be simply divided into Cloud Computing and Remote Sensing. At its easiest it refers to an infrastructure where the data storage, data accessing and the data handing out happen inside or from the cloud. CARMS applications move the computing power and data storage away from sensory devices and into the cloud, bringing applications and CARMS computing to not just smart-sensing users but due to a much broader range of sensing subscribers." This emphasizes that Remote Sensing benefits from Cloud Computing features-storage and data processing, also reveals a CARMS Services- moving part of the computation and the storage away from sensory devices. Thus, CARMS can be defined as: "An Infrastructure which provide Ubiquitous Computing with the help of Sensors, Clouds and *Internet* as a communication medium"

In CARMS, the previous or conventional ways of collecting and processing sensory data have been transferred to cloud and thus the limitation of remote sensing have been

Manuscript received September 10, 2015; revised December 9, 2015.

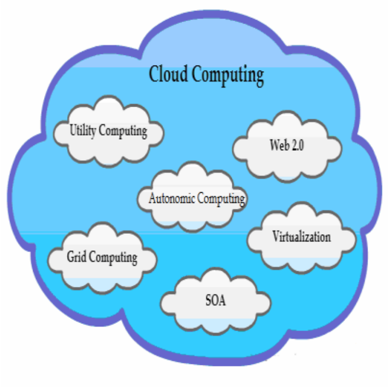
M. Zubair Khan is with Taibah University, P.O. Box 344, Madinah, Saudi Arabia (e-mail: mkhanb@taibahu.edu.sa).

Geeta Devi is with Invertis University Bareilly, India (e-mail: geetar01@gmail.com).

Yasser M. Alginahi is with Deanship of Academic Services, Taibah University, P.O. Box. 344, Madinah, Saudi Arabia (e-mail: yginahi@taibahu.edu.sa).

reduced, so the acquisition and processing mode of remote sensing applications have been totally changed. CARMS now enables:

1) Distributed sensory data collection: Sensed data can be collected from the distributed environment at one place.
2) Global resource and data sharing: Resources like sensors etc and sensed Information can be shared globally among the world.
3) Remote and real time data access: Sensed data can be accessed and analyzed in real time from anywhere.
4) Elastic resource provisioning and scaling: Where service users can provision and scale up and down their needed resources based on demand.
5) Pay-as-you-go pricing models: where cloud users can request, release, and pay for resources whenever needed.



+

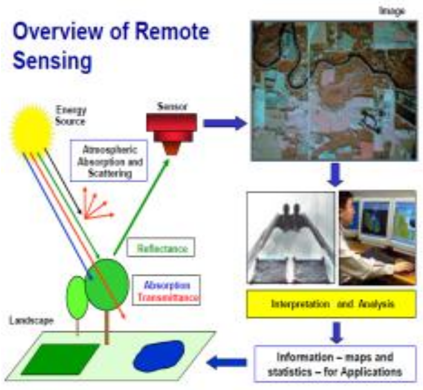

Overview of Remote Sensing
Image
Energy Source
Sensor
Atmospheric Absorption and Scattering
Reflectance
Absorption Transmittance
Landscape
Interpretation and Analysis
Information – maps and statistics – for Applications

Fig. 1. Cloud aided remote sensing architecture.

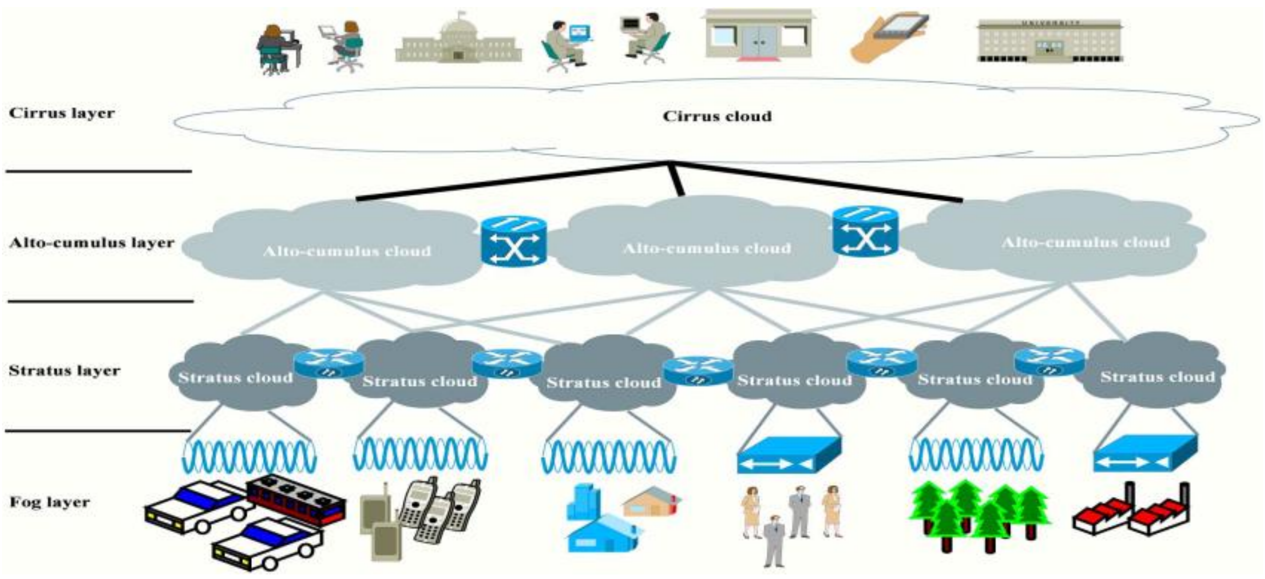


Fig. 2. Architecture of CARMS.

III. CARMS Existing Architecture

CARMS Architecture can be viewed as a geographically distributed platform that connects many billions of sensors and things and provides multitier layers of abstraction of sensors and sensor networks in Fig. 2. It has four main layers.

- Fog Layer
- Stratus Layer
- Alto-Cumulus Layer
- Cirrus Layer

A. *Fog Layer*

It encapsulates all physical objects, machines and anything that is equipped with computing, storage, networking, sensing and/or actuating resources, which can connect to, and be part of the *Internet*; for example: Smart Phones, Sensors, Vehicles etc. The sensory elements of this layer are those that collect and send raw sensed data to stratus layer or being pushed by fog layer to stratus layer.

The Functions of fog layer are to provide:

- Heterogeneous networking and communication infrastructures to connect billions of things.
- Unique identification of all things through *Internet* Protocol Version 6 (IPV6).
- Data aggregation points to serve as sensing clusters.

B. *Stratus Layer*

This is the second layer that consists of thousands of clouds whose main resources are sensory devices and SANS. Each Stratus cloud manages and acts as a liaison for a difficult group of SANS that share similar features, context, or properties. Stratus clouds are domain specific i.e. each cloud is very likely to be concerned with one application domain (e.g., Medical, environment, agriculture).

The Functions of stratus layer includes:

- Abstracting and virtualizing physical SANs through virtual network embedding (VNE) techniques.

- Handling and managing virtual SAN migration and portability across different clouds.
- Managing and ensuring operations and functionalities of virtual SAN instances.
- Enabling and managing (physical or virtual) SAN configuration to ensure network connectivity and coverage.
- Controlling the layer's operations and functionalities to ensure that customer's service level agreements (SLAs) communicated from higher layers are met.

This layer does not interact directly with CARMS customers, but serves them through requests received from higher layers.

C. Alto-Cumulus Layer

It is a middle layer that serves as a point of liaison between stratus and cirrus layers. It facilitates negotiations and SLA's between stratus and cirrus layers, and ensure that the agreed upon terms are not violated. An Alto-cumulus cloud may map to and organize multiple stratus clouds belonging to different domains. This enables inter-cloud resource sharing, thereby increasing resource elasticity and scaling.

The major functions of this layer are:

- Serving as a point of liaison between cirrus and stratus layer.
- Enabling business and payment transactions between cirrus and stratus layers by providing two-way brokerage services.
- Enabling and facilitating SLA negotiations between cirrus and stratus, and monitoring and ensuring that these SLAs are met. (Playing the role of Policy enforcement Agent).
- Coordinating and facilitating inter-cloud interactions, data exchange, task migration and resource sharing across different stratus clouds.

D. Cirrus Layer

This is the highest layer in the CARMS architecture. Its main role is to interact with CARS service customers and satisfy their requests with the help of lower layer. This layer does not deal with resource virtualization nor does it need to know which cloud handles which resources. What it needs to do is just to communicate customer's requests specified via their SLA to alto-cumulus clouds.

The Functions of this layer includes:

- Acting as a customer's entry point to CARMS systems.
- Allowing CARMS customers to set up their sensing task requirements and do whatever their chosen service model allows them to do (e.g., software configuration/ deployment).
- Providing online applications for remote data analysis to be used by customers to visualize their data in real time.

IV. CARMS Service Models

The main purposes of CARMS are to provide cloud customers with flexible access to data and sensing services, to allow them to develop their own domain-specific applications, and to allow clouds to share physical resources. Thus, the CARMS services are classified into three smart service models, which are analogous to Cloud Computing Service models:

A. SAIaaS

(Sensing and Actuating Infrastructure as a Service): In this design, Infrastructure provided are sensors or sensor networks. This model requires that physical sensor and SAN resources serve multiple sensing tasks concurrently. Customers can't make changes to physical resources (i.e., SANs & sensors), but have full control over their allocated virtual instances.

B. SAPaaS

(Sensing and Actuating Platform as a Service): In this model, CARMS customers are provided with a set of applications program interface (APIs) and libraries that they can use to develop their own sensing and actuating applications without worrying about the physical (SANs). These CARMS customers in this case, do have full control over their applications and can manage their resources, but they cannot alter any change in the physical or virtual infrastructure (e.g., shutdown a sensor, connect to another sensor, enable or deploy a new sensor etc).

C. SDAaaS

(Sensing Data and Analytics as a Service): Many practical applications need to have access and be able to process sensed data without needing to change anything in the physical sensors or in the virtual realization of SANs. CARS service customers using this service model, are only interested in the context in which sensed data is collected, its accuracy.

V. Description of Proposed Efficient Task Allocation Algorithm

CARMS is an *Internet* based technology in which sensing data are stored and accessed from the cloud. These stored sensing data at cloud are shared among its various users based on the type of sensing services they require. As the number of users increases at Cirrus Layer, the task to be schedule also increases and the performance of the CARMS system depends on the scheduling algorithm used in task allocation. The better scheduling algorithm can utilize better executing competence and maintain the load balancing of the system. Many researchers have already put many efforts to develop workload prediction techniques for cloud computing with the objectives of balancing workloads across servers which increases server's utilization and/or reducing power consumption. However, not much research has been done when it comes to cloud-based remote sensing. As the complexity of scheduling and task allocation of number of tasks or requests from cirrus layer increases, the ability to find good allocations and scheduling algorithm become more prominent. So the task allocation algorithm should be used for balancing and allocating customer request loads to optimize CARMS performance.

VI. Contribution

This part of our paper consists of a proposed Algorithm and its implementation.

A. Proposed ETA Algorithm

A task that enters into the System is equipped with the following information number of tasks (T) with their modules (m_i), number of processors (P_i) with their capacity IMCC (Inter Module Communication Cost): IMCC is a matrix which is used to represent the data communication between the modules during the execution and it can be calculated by calculating the amount of data transmitted from one module to another. Each entry of a IMCC between m_i and m_j of task T is represented by c_{ij}), which can be calculated as:

IMCC= (Communication between m_i on processor k and m_j on processor l)* (distance between processor k and l)

ETM (Execution Time Matrix): Matrix which determines the quantity of time taken by module to execute on a particular processor. And it depends on the capability or speed of the processor to which it is assigned and the amount of work to be performed by the module

With these given information our objective is to find an assignment of modules to processors, in a cloud, for which possible schedule is likely to be found by finding a suitable clustering and assignment.

Designing Algorithm for efficient task allocation models follows a number of systemic processes or steps which are as follows:

1) Each Layer on CARMS system (DSCS) consist of different servers or clouds say C_i (i=1 to n) for different Layers which in turns consist of, or maintain 'N' list of Virtual Machines/Processors (which may varies according to the demand of the system) where each of them may have different capacity or same, which is represented by 'M'.
2) Each cloud maintains different types of queues for different types of tasks (or modules). As each cloud is domain specific.
3) The users from different machines logs into the portal server at Cirrus Layer and send their request for different sensing services. These requests get collected into the Queue (Q) at Cirrus Layer. Let Q has 'n' number of task or request at time 't'.
4) Each task is transferred to further layers based on their type of sensing request.
5) 5. For each T_j of the layer is divided into different modules 'm_i'(i = 0 to n) and

CLUSTER_OF_MODULES( ); //based on their IMCC will be Executed.

6) Repeat (for all the Task say T_j in order).Then for cluster allocation the modules are assigned to the P_k based on the base case (Best Case, Average Case and Worst Case.

Check if (S_{ij}<=M_k) , where S_{ij} is the size of m_j of task T_i

If true

Assign modules of C_{ij} to Processor P_k.

7) Update processor table

M_k= M_k – S_{ij}

//M_k is the available memory capacity of processor k

// S_{ij} is the size of cluster C_{ij} of the task T and sort the P_k (by their capacity).

8) Repeat step 6 //for each cluster C_{ij} of T_j
9) Repeat steps 5 to 7 // for each layer (L_i)

Repeat until

Q = NULL.

10) Compute:
  - Total Cost (TC)
  - Processor Utilization (PU)
  - Response Time (RT)
  - Average PU (APU)

B. Implementation

The application for the above algorithm is developed in C and the learning has been carried out by using few examples given below, to judge the performance of the algorithm. Here, we assume that the IMC matrices, the execution time matrices, no_of_ tasks , no_of_modules in each task, no_of_processor, capacity of processors and size of the modules are given for every module of each task in units of time.

For example:

1) Given a task T_1, T_2 and T_3 with their modules and memory required (in MB) by each module.

T_1 = {m_{11} (4), m_{12} (5) , m_{13} (6)}

T_2 = {m_{21} (5), m_{22} (6), m_{23} (7), m_{24} (3), m_{25} (4)}

T_3 = {m_{31} (8), m_{32} (9), m_{33} (7)}

2) Number of processors say P_1, P_2 and P_3 with their Memory capacity say 100MB.
3) IMCC for task T_1 (Table I) and T_2 (Table II).

TABLE I: IMCC OF TASK T_1

	m_{11}	m_{21}	m_{31}
m_{11}	0	5	3
m_{21}	5	1	2
m_{31}	3	2	1

TABLE II: IMCC OF TASK T_2

	m_{21}	m_{22}	m_{23}	m_{24}	m_{25}
m_{21}	0	7	9	5	11
m_{22}	7	0	13	15	5
m_{23}	9	13	0	7	13
m_{24}	5	15	7	0	17
m_{25}	11	5	13	17	0

4) Execution time matrix of task T_1 (Table III).

TABLE III: EXECUTION TIME MATRIX OF TASK T_1

	p_1	p_2	p_3
m_{11}	2	5	3
m_{21}	5	1	2
m_{31}	3	2	1

Cluster formed for Task T_1 is as

C_{11} = {m_{11}, m_{21}}, Calculate the size of S_{11} = 4+6=10

C_{12} = {m_{31}}, Size of S_{12} = 9

Then after executing the proposed algorithm the following result will be shown (Table IV):

TABLE IV: EXECUTION RESULTS FOR PROPOSED ALGORITHM

Processor (P_k)	Memory Capacity (M_k)	Modules Assigned	Left Memory
P_1	100	m_{11}, m_{21}	90
P_2	100	m_{31}	91

Cluster formed for Task T_2 is as

C_{21} = {m_{24}, m_{25}}, Calculate the size of S_{21} = 3+4=7

$C_{22} = \{m_{22}, m_{23}\}$, Size of $S_{22} = 6+7 = 13$

$C_{23} = \{m_{21}\}$, Size of $S_{23} = 5$

After executing the second task the final table is shown in Table V.

TABLE V: RESULTS AFTER EXECUTING T_2

Processor (P_k)	Memory Capacity(M_k)	Modules Assigned	Memory Left
P_1	100	$m_{11}, m_{21}, m_{22}, m_{23}$	77
P_2	100	$m_{31}, m_{24}, m_{25}, m_{21}$	79

Table V, shows that how efficiently tasks are allocated and balanced load is obtained.

TABLE VI: TOTAL COST AND RESPONSE TIME FOR EACH PROCESSOR

Modules	Processors	PEC(k)	IPCC(k)	PEC(k)+ IPCC(k)
m_{11}, m_{21}	P_1	7	92	99
m_{31}	P_2	3	70	73

Table VI shows the total cost of each processor and the response time. The response time of the system in the above case is 99.

The simulation of the proposed algorithm was carried out using MatLab.

VII. PERFORMANCE EVALUATION

For scheduling/task allocation scheme I have compared the algorithm results with the base case I have developed. In order to show the efficiency of the design, I have compared the task scheduling scheme with the base case. The measure, I have used for performance comparison is Response Time.

The response time for increasing the number of tasks was found for the best case, the average case and the worst case. Fig. 3 shows the graph of Number of Tasks v/s Response Time (in sec) for proposed algorithm. In the best-case scenario, for the number of request the response time will not be affected. On the other hand when the number_of_tasks are more in worst case, we have taken the tasks with a higher frequency rate which causes the response time to increase rapidly. However, in average case, the response time for average case increases linearly with the number of tasks.

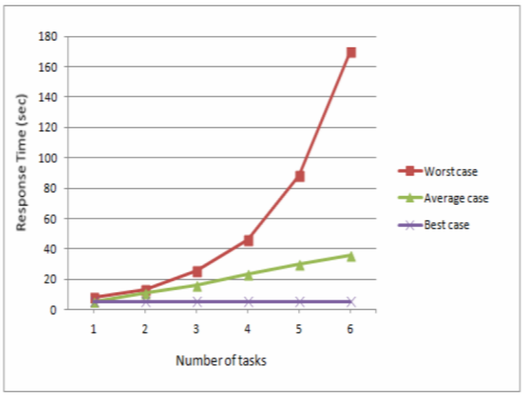


Fig. 3. Number of tasks / response time for task allocation.

VIII. CONCLUSION

As CARMS is a vast and new phrase of this technological era, so there is a huge scope of improvement in this area. Therefore, this paper provides an overview of CARMS system, algorithm or technique that allows selecting efficient task allocation algorithm. The theoretical analysis and simulation results which have been done show that the proposed algorithm has better performance both in finding solution and response time. I believe that the modeling technique and the algorithm presented in this paper are general, effective, time and cost efficient thus are applicable to practical CARMS system.

REFERENCES

[1] Nimbits Data Logging Cloud Sever. [Online]. Available: http://www.nimbits.com
[2] Pachube Feed Cloud Service. [Online]. Available: http://www.pachube.com
[3] IoT—ThingSpeak. [Online]. Available: http://www.thingspeak.com
[4] M. Whaiduzzaman, M. N. Haque, and M. R. Karim "A study on strategic provisioning cloud computing services," 2014.
[5] S. Abdelwahab, B. Hamdaoui, and M. Guizani, "Enabling smart cloud services through remote sensing: An internet of everything enabler" *IEEE*, 2014.
[6] J, Gubbi, R, Buyya, S. Marusic, and M. Palaniswami, *Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions*, Elsevier, 2013.
[7] M. Sanjay, V. Kumar, and R. Dalvi, "Sensor cloud: A cloud of virtual sensors," *IEEE Journal in Software*, IEEE, vol. 31, no. 2, pp. 70-77, Mar. 2014.
[8] S. Ehsan *et al*., "Design and analysis of delay-tolerant sensor networks for monitoring and tracking free-roaming animals," *IEEE Trans. Wireless Commun*., vol. 11, no. 3, pp. 1220–1227, Mar. 2012.
[9] X. Sheng, J. tang, X. Xiao, and G. Xue, "Sensing as a service: Challenges, solutions and future directions," 2013.
[10] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain., "A survey on sensor-cloud: Architecture, applications, and approaches," *International Journal of Distributed Sensor Networks*, p. 18, 2013.
[11] M. A. Tawfeek, A. El-Sisi, A. E. keshk, and F. A. Torkey, "cloud Task scheduling based on ant colony optimization," *IEEE*, 2013.
[12] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," College of Computer Science and Technology Jilin University, Chang Chun, China, 2011.
[13] W. Sun, N. Zhang, H. Wang, W. Yin, and T. Qiu, "PACO: A period ACO_based scheduling algorithm in cloud computing," School of Software Technology Dalian University of Technology Dalian, China, 2013.
[14] S. Selvarani and G. S. Sadhasivam, "Improved cost-based algorithm for task scheduling in cloud computing," Coimbatore, India, 2010.
[15] M. A. M. Viera *et al*., "Scheduling nodes in wireless sensor networks: A Voronoi approach," presented at International Conference on Local Computer Networks, IEEE, 2003.
[16] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing," presented at 13th Int'l Conf. Network-Based Information Systems (NBIS 10), IEEE CS, 2010.

M. Zubair Khan got the Ph.D. degree in computer science and information technology from Faculty of Engineering, M.J.P. Rohilkhand University, Bareilly India, and the master of technology in computer science and engineering from U.P. Technical University, Lucknow, India.

He is currently working as senior faculty member in the Department of Computer Science, Deanship of Academic Services, Taibah University. Past he has worked as head and associate professor, in the Department of Computer Science and Engineering, Invertis University, Bareilly India. He has published more than 36 journals and conference papers.

Dr. M. Zubair Khan is a member of Computer Society of India since 2004. His current research interests are data mining, big data, parallel and distributed computing , theory of computations, and computer networks. He has more than 13 years teaching and research experience.

Geeta Devi is currently working as an assistant professor in ANA College of Management and Technology, Bareilly (India). Her research interest includes cloud computing, cloud scheduling, data mining and scheduling algorithms. She obtained the Bsc (computer Sc) during 2008, the Msc (computer Sc) during 2010 from Pune University, Maharashtra (India) and the M.Tech. degree in computer science and engineering in 2015 from Invertis University, Bareilly, Uttar Pradesh (India). She has total 3 year experience of teaching in different colleges of Pune University and She has published some technical papers in conference proceedings.

Yasser M. Alginahi earned a Ph.D. in electrical engineering from the University of Windsor, Ontario, Canada, and a master of science in electrical engineering from Wright State University, Ohio, U.S.A.

He is an associate professor at the Department of Computer Science, Deanship of Academic Services, Taibah University. He is also the consultation unit coordinator at the IT Research Center for the Holy Quran and its sciences, Taibah University. Dr. Alginahi is on editorial board of several international journals. He published a book entitled *Document Image Analysis* and he has published over 60 journal and conference papers.

Dr. Alginahi is a licensed professional engineer, Ontario, Canada, a member of Professional Engineers Ontario, a senior member of IEEE and IACSIT since 2010. He worked as a principal investigator and co-principal investigator on many funded research projects by the Deanship of Scientific Research at Taibah University and other organizations such as King Abdul-Aziz City of Science and Technology. His current research interests are big data, information security, document image analysis, pattern recognition, ocr, modeling and simulation, and numerical computations.