Service Level Agreement based Scheduling Techniques in Cloud: A Survey

Rajeshwari B.S.
Assistant Professor, Dept. of Computer Science and Engineering
B M S College of Engineering
Bangalore, India

M. Dakshayini, PhD
Professor, Dept. of Information Science and Engineering
B M S College of Engineering
Bangalore, India

H.S. Guruprasad, PhD
Professor and Head, Dept. of Computer Science and Engineering
B M S College of Engineering
Bangalore, India

ABSTRACT
Nowadays enterprises need to maintain plenty of applications accessing by millions of users all over the world. Maintaining their own infrastructure, managing software requirements and handling excessive internet traffic is difficult. This makes them to move towards cloud computing. Cloud computing is a service provisioning technique, where customers can rent any resources like hardware, software, or platform to develop an application. Customers need to pay only for how much the resources were consumed, can dynamically increase or decrease the resource capacity as needed. Because customers are paying for the services, they expect quality of service from the provider. Providing a quality of service and attracting the customers is a challenging issue for the providers. If not, customers will move towards other cloud providers. Thus Service Level Agreement (SLA) is made between providers and customers that include service quality, resources capability, scalability, obligations and consequences in case of violations. Satisfying SLA is very important and a challenging issue. In this paper, different framework and techniques proposed by the different authors for providing a quality of service and maintaining SLA are discussed.

Keywords
Cloud Computing, Quality of Service, Service Level Agreement.

1. INTRODUCTION
Cloud computing is a service provisioning technique consisting of various data centers distributed all over the world connected through WAN. Each datacenter consists of multiple physical machines connected by means of LAN. In each physical machine multiple virtual machines were created, sharing the hardware and storage resources. Different applications run over on each virtual machine. Users can access these applications through web portal. Cloud computing has four distinct features [6]:

- **It is elastic:** user can increase or decrease the resources as needed.
- **Pay per use:** Usage is metered and user pays only for how much the resources were used.
- **Operation:** The services are completely handled by the provider.
- **Self-service:** Users can operate through the console to add a new CPU, a server instance or extra storage.

2. SERVICE LEVEL AGREEMENT
In this cloud scenario, there will be a service providers who are providing service and customers who are using services. Customers pay for the resources depending on the usage. Since customers pay for the services, they expect quality of service from the provider. Thus there will be a clause between providers and customers in providing a Quality of Service (QoS) called as Service Level Agreement (SLA). SLA not only includes functional requirements like service quality, resource capability, and scalability but also includes non-functional requirements like security, privacy, trust etc. Retaining the customers is challenging for the providers. By providing quality of service and maintaining SLA, customers can be retained. Thus different authors' proposed different frameworks and SLA based techniques used for maintaining SLA, avoiding SLA violations. According to an agreement, if SLA violates providers may lose customers. Thus by continuous monitoring of customer services and taking appropriate action ahead of time avoids agreement violations and can retain customers. SLA based scheduling includes scheduling task to the appropriate machine based on SLA, monitoring of resource usage, customer service execution status, periodically checking for SLA violation. SLA mainly includes

i) Responsibilities of both providers and customers.

ii) List of services and its description that is being provided to the customer by the provider.

iii) Agreement on functional and nonfunctional requirements provided by the provider.

iv) Legal context that has been negotiated by the provider and customers.

The SLA is split into the different stages, each addressing specific set of customers for the same services, in the same SLA. [22]

• **Corporate-level SLA:** Service level management issues are related to every customer throughout the organization. These issues are likely to be less volatile and so SLA reviews are not required frequently.

• **Customer-level SLA:** Service level management issues are related to the particular customer group, regardless of what services being used.

• **Service-level SLA:** Service level management issues are related to the specific services, in relation to this specific customer group.

3. SERVICE LEVEL AGREEMENT BASED SCHEDULING TECHNIQUES

Stefano Ferretti et al., [4] proposed an architecture that supports SLA driven configuration, management and optimization of cloud resources and services. The architecture mainly includes configuration service and load balancer. In turn load balancer includes monitoring services and SLA policy engine. Load balancer is responsible for dispatching request and balancing load among the servers. It takes request from clients and assigns these requests to an appropriate platform and balancing load among them. Monitoring service within load balancer monitors the incoming request and its response time so as to check whether a SLA associated with the request are satisfied or violated. SLA policy engine within load balancer analyses the logs prepared by monitoring service and finds whether system configuration need to be changed or not. If necessary, invokes configuration service module to reconfigure the resources. If SLA is fulfilled according to an agreement and platform resources results unused, platform reconfiguration occurs to release the unused resources. If SLA is violated, then platform reconfiguration occurs to add in additional resources to the platform. Proposed strategy optimizes resource utilization by dynamically adding and releasing number of VM's to an application platform depending on load, while honoring SLA but incurs additional overhead. The downside of the strategy is as load imposed, adding number of VM's in execution environment exceeds certain limit, forces some scalability problem for distributed application such as shared DB.

Suneel K S et al., [7] presented an approach to monitor SLA compliance at the cloud service provider that can be implemented at the client end without need of third party. Using this approach, the cloud users can continuously monitor the SLA compliance at the cloud end. This proposed approach includes mainly two functions **i)** Information Fetch Task Generator function **ii)** Evaluator function. Information Fetch Task Generator function, by taking SLA as input, generates information fetch task. This information fetch task will be sent along with the requested task to the cloud for execution. In the cloud, for the set of task hash value is calculated and generates log file that contains hash value of the task and their arrival time. After getting result from the cloud, evaluator function evaluates the SLA breach at the cloud. If percentage of SLA breach is greater than the acceptable threshold, then raises a SLA breach notification at the cloud user end .This is a good effort where information fetch task is sent along with the set of tasks and using the result of information fetch packet returned from the cloud, percentage of SLA breach can be identified, but the time at which information fetch task is executed in the cloud is very important in identifying the percentage of SLA breach.

Xiaomin Zhu et al., [15] proposed a **"QoS aware fault tolerant scheduling algorithm called QAFT"**. The proposed algorithm can tolerate failure of one processor at a time for real time tasks that wants QoS in heterogeneous systems. The algorithm starts execution of primary copies first and delay execution of backup copies. By making passive execution of backup copies, avoids execution of both primary copies and backup copies simultaneous incorporate effective resource utilization. The proposed model also provides higher guarantee ratio, adaptively adjust the QoS levels of incoming tasks depending upon the system load and avoids task rejection. When the system is in heavy load, then QoS levels of accepted tasks are degraded in order to increase guarantee ratio. When the system is in light load, then QoS levels of accepted tasks are increased in order to provide high quality service. Hence in conclusion, proposed strategy by adaptively adjusting QoS levels of real time systems based on system load increases guarantee ratio, QoS, reliability and optimum resource utilization in heterogeneous systems with tolerance of only one processor's permanent failure at one time instant for real time tasks with QoS needs.

Ahmed Amamou et al., [19] proposed an algorithm **"SLA based Dynamic Bandwidth Allocator (DBA)"**.The proposed DBA algorithm allocates bandwidth to the application based on established SLA agreement. The algorithm continuously monitors bandwidth allocated for each application environment and dynamically adjusts the bandwidth. The architecture consists of a special virtual machine called driver domain and multiple virtual machines, each hosting different applications. Virtual machines are grouped based on priority into different classes. Each virtual interface $\mathbf{v_if_i}$ are connected to physical interface P. DBA algorithm browses each virtual interface $\mathbf{v_if_i}$ and measures the bandwidth $\mathbf{B_i}$ currently using by each virtual machine and adjust accordingly. **i)** If virtual machine is using bandwidth within the range, then algorithm does not perform any change. **ii)** If virtual machine is using bandwidth above the specified range, then readjust bandwidth to maximum range and remaining bandwidth will be added to available bandwidth $\mathbf{B_x}$. **iii)** If virtual machine is using bandwidth below the specified range, then algorithm finds if any bandwidth available. If so, bandwidth will be added and readjust to minimum range. If not, then algorithm readjusts the bandwidth of all virtual machines belonging to the same class and will be added to the virtual machine, readjust to minimum range. The proposed method also optimizes the physical resource usage of CPU and memory by dropping packets at virtual machine directly instead of driver domain. This avoids transferring packets to destined virtual machine from driver domain through I/O channel, saves memory cycles and CPU cycles. Good effort, sharing of bandwidth among VM's depending upon the traffic honors SLA, reduces the packet loss rate. But the model is focusing only on bandwidth, hence application performance may be affected by other resources such as CPU processing, memory and storage.

Rajeshwari B S et al., [21] proposed a framework "**Optimized Service Level Agreement Based Workload Balancing Strategy in Cloud Environment**". The presented framework offers both balancing the load among the servers as well as QoS in cloud. In this framework, depending upon the servers processing capacity, the servers are grouped into 3 clusters as **i)**High processing power servers' cluster **ii)**Medium processing power servers' cluster **iii)**Low processing power servers' clusters. The framework comprises two scheduling algorithms **i)** SLA Based Scheduling algorithm **ii)** Idle-Server Monitoring algorithm using at two different stages. At the first stage, when a task enters into the task queue, SLA based scheduling algorithm based on task length, deadline to finish task and cost paid by the user computes the priority of the task. Depending upon the calculated priority, algorithm schedules the task to the respective servers cluster. At the second stage, within each cluster Idle-Server Monitoring algorithm monitors a set of servers in its cluster. When it receives a task, algorithm checks for any idle server in its cluster. If found, it assigns a task to the identified server. If not, Idle-Server Monitoring algorithm running within medium processing power servers cluster finds for any server is idle in high processing power server's cluster. If found, it assigns its task to the identified high power machine. Correspondingly, Idle-Server Monitoring algorithm within low processing power server's cluster finds any idle server in medium

processing power servers' cluster. If found, it assigns its task to the identified medium power machine. By doing this, the high power machines and medium power machines are utilized effectively, provide good response time, reduce waiting time of the task and achieves better load balancing among the servers. In this work, easy to manage by clustering of servers based on processing and memory capacity. By moving medium priority tasks to idle high processing servers and low priority tasks to medium processing servers, high power and medium power machines are utilizing efficiently. But the downside of the proposed strategy is that it lacks monitoring scheme for catching QoS violations.

Attila Kertesz et al., [13] presented architecture for SLA based resource virtualization for executing user applications in cloud. The architecture mainly includes three modules **i)** Agreement Negotiation module **ii)** Service Brokering module **iii)** Service Deployment module. Agreement Negotiation module is responsible for negotiation on SLA between users and providers, determines user QoS parameters, rewards and penalties for meeting and violating them. Architecture includes components such as Meta Negotiator (MN), Meta Broker (MB), Broker (B), Automatic Service Deployment (ASD), Physical and virtual resources. During agreement negotiation process, the negotiation takes place between users→MN, MN→MB, MB→B, B→ASD. Once SLA is negotiated between user and provider, Service Brokering module finds the required services with the help of ASD. Service Brokering module is responsible for service discovery, match making, interactions with information system, service registers and repositories. If the required service is not found, it calls Automatic Service Deployment module to install the required service on the specifically selected resource dynamically and starts executing.

Al Amin Hossain et al., [2] proposed a Cloud Brokage model that boost up customer satisfaction and diminish cloud service provider's anxiety for continuing their business. Presented model consist of cloud broker, different cloud service providers to attain cloud service for a particular cloud customer. Cloud broker is a negotiator between customers and multiple service providers. Customer request for their service to cloud broker. Cloud broker searches for compatible service provider and assigns to the requested customers. While receiving service, broker observes amount of utilization of registered resources and QoS. If customers request to discontinue the remaining service, broker calculates refundable amount for unutilized resources, service quality degradation, service cancellation and profit obtained. Thus architecture ensures SLA by providing refundable service in case of unused resource, service quality degradation and service cancellation. Hence proposed model overcomes drawback of pay-as-you-go pricing model in terms of fairness by refunding for SLA violations, service quality degradation, service cancellation and unutilized resources.

Vincent C et al.,[1] presented a novel scheduling strategy that schedules the incoming requests on virtual machines based on agreed SLA by considering many SLA parameters such as CPU requirement, memory, storage and network bandwidth. The proposed model includes on demand resource allocation strategy that automatically creates new virtual machines when an appropriate virtual machine is not available for application deploy. The proposed method also includes a load balancer for effective distribution of applications execution on the cloud resources. On demand resource provisioning strategy is evaluated by deploying applications under three different circumstances **i)**The first situation deals with only the deployment of web applications service request **ii)**Second situation deals with deployment of only high performance computing (HPC) applications service request **iii)**Third situation deals with deployment of web applications and HPC applications. The strategy obtains better resource utilization and deployment efficiency for any kind of applications as compared to fixed resource provisioning technique.

Xiao Liu et al., [3] proposed a standard QoS framework for cloud workflow systems. The proposed framework mainly consists of four modules **i)**QoS Requirement Specification Module **ii)**QoS Aware Service Selection Module **iii)**QoS Consistency Monitoring Module **iv)**QoS Violation Handling. Model is implemented in a consecutive fashion in order to provide QoS for cloud workflow instances in 3 stages **i)** During modeling stage, real world e-business or e-science processes are modeled which contains process structure, task definition for number of workflow activities and QoS requirement such as performance, reliability and security. The provider will negotiate with their customers. **ii)** During instantiation stage, cloud workflow system searches for cloud service both functional and non-functional QoS requirement that satisfies the execution of workflow activities. **iii)** During execution stage, workflow execution engine will manages data and control flows, performs runtime management such as monitoring and exception handling mechanisms that ensures detection and recovery of QoS violations such that service negotiated agreement be successfully fulfilled. Good effort, all local temporal violations are handled automatically and make global SLA violation rate kept very close to 0%.

Kun Ma et al., [5] proposed a lightweight, scalable framework which incorporates some open source monitoring tools. These monitoring tools perform end to end measurements of softwares and virtual machine instances in the public cloud. These monitoring tools monitors QoS parameters at Infrastructure as a Service (IaaS) layer and Software as a Service (SaaS) layer without modifying the implementation of the monitored objects. The proposed framework mainly integrates 7 open source monitoring tools that periodically monitors IaaS layer and SaaS layer, gathers runtime information and sends UDP packet to the management. At the bottom, it monitors infrastructures such as utilization of CPU, memory, disk, network etc on both physical and virtual machines and hypervisor that manages guest OS. On the top of infrastructures, it monitors softwares and services which provides an appropriate runtime environment for applications. The architecture also has manager-agent and module centralized component that improves the performance of application in public cloud based on collected runtime information. Hence in proposed strategy, by integrating monitoring tools, cpu, memory, network, storage usage can be identified, load on physical and VM machines, availability, security, vulnerabilities of a website, software service can be monitored, thus helps in improving overall performance of an applications in public cloud.

Mario Mac´ıas et al., [8] discussed 7 business rules for revenue maximization of cloud providers. It encompasses the automatic enforcement of Business Level Objective (BLO) by means of bi-directional data flow between market and resource layers. The framework incorporates integrated set of policies that work together for revenue maximization of cloud provider along with the performance concerns. The 7 business rules discussed that maximizes revenue of providers includes 1)Dynamic Pricing 2)Resource Over Provisioning 3) Selective SLA Violation 4) Selective SLA Cancellation 5) Ranges for

QOS 6) Tasks Reallocation and 7) Redistribution of Assigned Resources.

Xiaomin Zhu et al., [9] proposed a "**QoS based self-adaptive scheduling algorithm called QBSA**". The presented algorithm improves QoS of real time applications that are running on heterogeneous clusters based on the current system load. When the system is in heavy load, the QBSA algorithm degrades the QoS level of new arrival tasks or tasks waiting in the local queue of nodes to guarantee high schedulability. In contrast, when the system is in a light load algorithm increase QoS level for the newly arrived task in order to provide higher quality service by an effective utilization of system resources. Thus the proposed algorithm provides good schedulability and higher guarantee ratio, when the system is in heavy load i.e., avoids task rejection and provides higher quality service and effective resource utilization when the system is in light load. Schedulability is the main goal of the algorithm when numbers of nodes are less or tasks arrive quickly. The proposed methodology is advantageous that when number of nodes is more, algorithm improves overall performance of a system in terms of both guarantee ratio and QoS of all accepted tasks. But downside, when number of nodes is less, algorithm provides higher guarantee ratio, but comparatively not strives to maximize QoS of all accepted tasks.

Hien Nguyen Van et al.,[11] presented an automatic resource management system that provides automatic dynamic provisioning and placement of virtual machines based on SLA and resource cost. The presented method has a capacity to "scale up" by increasing more resources to a single server or "scale out" by increasing more servers for an application. In the proposed approach, set of pre-defined virtual machine classes were created and the algorithm chooses virtual machines from the pre-defined set for an application. Each virtual machine class was defined with a specific CPU and memory capacity. The proposed framework mainly contains two modules **i)** Local Decision Module (LDM) **ii)** Global Decision Module (GDM). LDM is connected with each application environment (AE), evaluates whether need to allocate more virtual machines or releasing existing virtual machines from AE depending upon the current workload and SLA metric received from application specific monitoring probes. Then all LDM's interact with GDM. GDM by receiving a utility function from every LDM's and system performance of virtual machines and physical machines ,suggests to hypervisor either to add new virtual machine to an AE, upgrade virtual machines to next class of virtual machines, degrades virtual machines to previous class of virtual machines or live virtual machines migration. Then GDM notifies LDM. Hence in proposed strategy, by collecting local and global utilization of resources dynamically add and removes VM's on physical machines satisfies both SLA and optimal placement of VM on physical machine, reduces energy consumption

Asma Al Falasi et al., [12] presented SLA monitoring model designed for federated cloud environment. The proposed model administers social relationships established between different cloud services. In the paper, the author discussed lifecycle of SLA by taking sky framework as an example of federated cloud environment. Sky framework is a collection of interconnected cloud services from different cloud vendors forming. The proposed model manages multi-level SLA's, monitors socially interconnected cloud services, detects all SLA violations and communicates these violations to the concerned providers. The proposed architecture mainly includes two modules **i)** Socialization module **ii)** Federation Module. Socialization module is responsible for **i)** storing and manipulating data pertaining to cloud services **ii)** implementing sky business model **iii)** managing membership, enforcement of rules, finding of violations and deciding on penalties. Federation module is responsible for **i)** receiving requests from sky broker, performing agreement between provider and users, forwards the required information to the monitoring manager to monitor run time operations **ii)** maintains information regarding each application and allocated resources **iii)** Monitor manager module checks for SLA violations, ensuring that SLA is honored. The advantage of the proposed architecture is that by administering and monitoring SLA at different levels such as uploading SaaS, searching SaaS, downloading SaaS, creating folder SaaS, broker detects QoS violations and sends report to the concerned cloud providers to take future actions. Thus manages the social relationship between the cloud services in federated cloud and improves service performance.

Asma Al Falasi et al., [14] proposed a framework that provides dynamic specification of SLA, verification and composition of services depending on SLA in the cloud using genetic algorithm. The framework mainly composed of three components **i)**Third Party Cloud Directory **ii)** Cloud Providers **iii)**Trusted Broker. Third party cloud directory acts as an intermediator between requestors and providers, acts like yellow pages. In this, providers offering their infrastructures as a service. They expose their infrastructure as web services for clients. They offer different service plans with different QoS range like Bronze, Gold, Silver where each plan is a specific service that needs to be composed to carry out tasks. Broker module includes two modules **i)** Web Service Verifier **ii)** Composition Engine. Web service verifier tests concrete services for QoS parameter specified by requestor. Composition engine decides on what concrete services of different QoS range needs to be composed to satisfy client's SLA. In this framework, first client searches repository of cloud service providers and give their QoS parameters. Then cloud provider looks up services, QoS parameters specified by the user and communicate with a list of service API's to a trusted broker. Broker performs few verification tests on the specified services to validate the QoS parameters. Only the services that passed the verification tests are given as input to the composition engine. Composition engine decides on which specific services to be included in the composition in order to satisfy specified SLA. Finally resultant SLA is negotiated back to the client, either to agree on or select an alternate solution or to move on to the other cloud provider. The proposed framework helps to delegate the process of monitoring the cloud service QoS measures to a trusted broker, thus reducing load on providers.

Ivona Brandic et al., [16] presented an approach "**A Layered Approach for SLA-Violation Propagation in Self-Manageable Cloud Infrastructures (LAYSI)**". Proposed layered architecture maps low resource metric like down time, uptime and currently available storage into the SLA parameters like service availability periodically and propagates possible future SLA violation to an appropriate layer. The algorithm finds the future SLA violation by defining more restrictive threshold called Threat Threshold than the SLA violation threshold. The architecture includes five layers like Meta Negotiator (MN), Meta Broker (MB), Broker (B), Automatic Service Deployer (ASD) and services and resources. First user starts negotiation with the MN with specific QoS parameters. MN than calls MB. MB compares the requirements specified by the user with the available brokers. If found, accepts and performs negotiation else offers

for renegotiation. This process continues until both sides agree. If agree for negotiation, user calls for execution of service with a specific Service Description that contains image of OS, middleware, application, data, configuration along with SLA. Broker executes service if available otherwise calls ASD to deploy the service and execute. The architecture uses knowledge database to take a decision based on past experience to avoid future SLA violation. It looks for similar cases stored in the database periodically and uses the solution of that particular case in order to avoid possible SLA violation, stores that experience in the knowledge database probably to be useful for future problem solving. If broker not find solution, then propagates failure to the ASD possibly for virtual machine migration. This is a good effort, by identifying SLA's that might be violated in future, uses knowledge database, retrieve most similar case, and reuse the solution that has taken up, avoids future SLA violations almost to 0%. But overhead in maintaining knowledge database and case based reasoning

Rajkumar Buyya et al., [17] discussed vision, challenges and architectural elements of SLA based resource management. The proposed architecture supports integration of market based provisioning policies and virtualization technologies in order for the flexible allocation of resources to an application. The architecture mainly includes 4 main entities i**)** Users/Brokers **ii)** SLA Resource Allocator **iii)** Virtual Machines and **iv)** Physical Machines. Users interact with cloud management system through broker. SLA resource allocator acts as an interface between users and the infrastructure. SLA resource allocator includes **1)** SLA request examiner, which understands QoS requirements given by the user and determine whether to accept or reject. **2)** Pricing component charges to service requests and manages computing resources by prioritizing resource allocation.**3)** Dispatcher component deploys application on appropriate virtual machine; create virtual machine image and starts on selected physical machine.**4)** Virtual Machine and Application Monitor component, keep tracks of availability of virtual machines and their resource capacities, monitors the performance continuously in order to identify whether any breach in SLA and notifies resource allocator.**5)** Accounting and SLA management component maintains actual usage and calculates final costs. Multiple virtual machines are started and stopped on physical machines dynamically depending upon the requests. The proposed strategy handles dynamic nature of cloud very well by adding and allocating resources for the tasks depending upon deadlines, meets deadline for all incoming tasks with resource cost optimization to a larger extent compared to static allocation. But overhead in finding number of servers to be added based on deadline of incoming tasks.

K C Gouda et al., [18] presented priority based resource allocation algorithm. The algorithm takes different parameters like time, processor requests, importance and cost for each task. Time is the execution time needed to finish the task. Processor request is the number of CPU's needed to execute the task. Importance is how important the user is to the cloud administrator and cost paid is amount paying by the user to the cloud administrator. The proposed algorithm for each tasks, computes the node priority and time priority value based on the specified conditions. It checks whether total number of requested nodes is less than the currently available nodes. If yes, schedules the task. If request cannot be satisfied with the currently available nodes, then those tasks are put into the queue. Finally the queue is sorted based on the priority and begins executing the task from the beginning of the queue. If the requested resource exceeds the limit, then those tasks will be rejected. Good effort, algorithm by calculating priority based on some parameters helps in proper scheduling of tasks to an appropriate server provides sustained performance of different application running on different configuration servers and optimizes resource allocation, but algorithm does not take care of task rejection.

Xiaomin Zhu et al., [10] proposed a "**Self-Adaptive QoS Aware Scheduling Algorithm SAQA**" for soft real time applications that are running on heterogeneous clusters.The proposed algorithm defines QoS level range for each incoming tasks. When the system is in a less load, the algorithm increases the QoS level for all incoming tasks within the specified range to provide high QoS by utilizing the system resources efficiently. When the system is in heavy load, the algorithm decreases the QoS level of incoming tasks as well as tasks waiting in the queue within the specified range in order to provide high load distribution and avoids task rejection. i.e., High load distribution and avoiding task rejection is the major goal when the system is in the heavy load and providing quality service is the major goal when the system is in a light load by effective utilization of the system resources. The proposed SAQA algorithm uses the techniques of two algorithms: non-preemptive EDF (Earliest Deadline First) algorithm and RF (Response First) algorithm that considers both deadline as well as a response time of the task into the account while scheduling. Thus proposed algorithm considers deadline and quality service needs into account for scheduling independent, aperiodic, soft real time applications running on heterogeneous clusters.

Rajkumar Buyya et al., [20] discussed vision, challenges and architectural elements of Intercloud: federation of cloud computing environments. Presented federated cloud environment supports scaling of applications across multiple cloud providers. Architectural framework mainly consists of 3 components **i)** Exchange **ii)** Cloud Coordinator **iii)** Cloud Broker. Cloud Exchange component acts as a market maker, brings together several cloud providers and consumers. It aggregate infrastructure requests from the brokers and evaluates them against currently available infrastructure published by cloud coordinators. Cloud coordinator service component specific to each cloud manages a domain specific enterprise cloud and its membership to the overall federation which is driven by market based trading and negotiation protocols. Cloud Coordinator component exports services of the cloud to the federation includes components like scheduling and allocation, market and policy engine, application composition engine, virtualization, sensors, discovery and monitoring. Cloud Broker component selects suitable cloud service providers for the requested users according to their requirements through cloud exchange and negotiates with cloud coordinators for assignment of resources that meets QoS needs specified by the users. This is a good effort of finding a perfect match between provider's capability and customer's requirements.

4. CONCLUSION

Currently cloud computing has created a big trend, as users no need to maintain hardware, software, storage. Users can takes all services from the cloud as and when needed. Since more number of users are attracting towards cloud, providing QoS to the users is challenging task for the cloud service providers. SLA is made between the cloud service providers and users in providing quality service. By avoiding SLA violations, cloud users enjoy better services from the provider. This will also help provider to improve his market reputation and revenue.

In this paper, various SLA based scheduling strategies proposed by different authors in satisfying the agreed SLA are discussed. Different architecture that take care of SLA violations by considering multiple parameters such as CPU requirement, memory, storage, network bandwidth and fault tolerance, avoiding SLA violations by using open source tools, SLA violations on federated cloud environment are discussed.

5. ACKNOWLEDGMENTS

The work reported in this paper is supported by the college through the TECHNICAL EDUCATION QUALITY IMPROVEMENT PROGRAMME [TEQIP-II] of the MHRD, Government of India.

6. REFERENCES

[1] Vincent C. Emeakaroha, Ivona Brandic, Michael Maurer, Ivan Breskovic, "SLA-Aware Application Deployment and Resource Allocation in Clouds", Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual, Munich 18-22 July 2011, PP:298-303,PrintISBN:978-1-4577-0980-7,DOI:10.1109/ COMPSACW.2011.97.

[2] Al Amin Hossain, Eui-Nam Huh, "Refundable Service through Cloud Brokerage", IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, June 28 2013-July 3 2013, PP: 972–973, Print ISBN:978-0-7695-5028-2, DOI:10.1109/CLOUD.2013.115.

[3] Xiao Liu , Yun Yang, Dong Yuan, Gaofeng Zhang, Wenhao Li, Dahai Cao, "A Generic QoS Framework for Cloud Workflow Systems", Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing, Sydney, NSW, 12-14 Dec. 2011, PP: 713 – 720, PrintISBN:978-1-4673-0006-3,DOI:10.1109/DASC.2011. 124.

[4] Stefano Ferretti, Vittorio Ghini, Fabio Panzieri, Michele Pellegrini, Elisa Turrini, "QoS–aware Clouds", IEEE 3rd International Conference on Cloud Computing, Miami, FL, 5-10 July 2010, PP: 321 – 328, Print ISBN:978-1-4244-8207-8, DOI:10.1109/CLOUD.2010.17.

[5] [5] Kun Ma, Runyuan Sun, Ajith Abraham, "Toward a lightweight framework for monitoring public clouds", Fourth International Conference on Computational Aspects of Social Networks (CASoN), Sao Carlos, 21-23 Nov. 2012, PP: 361 – 365, Print ISBN:978-1-4673-4793-8, DOI:10.1109/CASoN. 2012.6412429.

[6] Rajeshwari B S, Dr. M Dakshayini, "Comprehensive Study on Load Balancing Techniques in Cloud", an International Journal of Advanced Computer Technology, Volume 3, Issue 6, June 2014, PP: 900-907, ISSN: 2320-0790.

[7] Suneel K S, Dr. H S Guruprasad, "A Novel Approach for SLA Compliance Monitoring In Cloud Computing", International Journal of Innovative Research in Advanced Engineering, Volume 2, Issue 2, February 2015, PP: 154-159, ISSN: 2349-2163.

[8] Mario Mac´ıas, J. Oriol Fito , Jordi Guitart, "Rule-based SLA Management for Revenue Maximisation in Cloud Computing Markets", Proceedings of the 6th IEEE/IFIP International Conference on Network and Service Management (CNSM). 2010, PP: 354-357, ISBN: 978-1-4244-8908-4.

[9] Xiaomin Zhu, Peizhong Lu, "A QoS-Based Self-Adaptive Scheduling Algorithm for Real-Time Tasks on heterogeneous clusters".

[10] Xiaomin Zhu, Jianghan Zhu, Manhao Ma, and Dishan Qiu, "SAQA: A Self-Adaptive QoS-Aware Scheduling Algorithm for Real-Time Tasks on Heterogeneous Clusters", 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, PP: 224-232, ISBN: 978-0-7695-4039-9 DOI: 10.1109/CCGRID.2010.64.

[11] Hien Nguyen Van, Frederic Dang Tran, Jean-Marc Menaud, "SLA-aware Virtual Resource Management for Cloud Infrastructures", IEEE Ninth International Conference on Computer and Information Technology, Xiamen,11-14 October 2009,PP: 357 – 362, ISBN:978-0-7695-3836-5,DOI: 10.1109/CIT.2009.109.

[12] Asma Al Falasi, Mohamed Adel Serhani, Rachida Dssouli, "A Model for Multi-levels SLA Monitoring in Federated Cloud Environment", 2013 IEEE 10th International Conference on Ubiquitous Intelligence & Computing and 2013 IEEE 10th International Conference on Autonomic & Trusted Computing, Vietri sul Mere, 18-21 December 2013, PP: 363 – 370, ISBN:978-1-4799-2481-3, DOI:10.1109/UIC-ATC.2013.14.

[13] Attila Kertesz, Gabor Kecskemeti, Ivona Brandic, "An SLA-based Resource Virtualization Approach For On-demand Service Provision", 3rd international workshop on Virtualization technologies in distributed computing, 15th June, 2009, PP: 27-34, ISBN: ACM 978-1-60558-580-2 DOI: 10.1145/1555336.1555341.

[14] Asma Al Falasi, Mohamed Adel Serhani, "A Framework for SLA-Based Cloud Services, Verification and Composition", International Conference on Innovations in Information Technology, Abu Dhabi, 25-27 April 2011, PP: 287–292, ISBN: 978-1-4577-0311-9, DOI: 10.1109/ INNOVATIONS.2011.5893834.

[15] Xiaomin Zhu, Jianghan Zhu, Manhao Ma, Dishan Qiu, "QAFT: A QoS-Aware Fault-Tolerant Scheduling Algorithm for Real-Time Tasks in Heterogeneous Systems", 13th IEEE International Conference on Computational Science and Engineering, Hong Kong, 11-13 December 2010, PP: 80 – 87, ISBN: 978-1-4244-9591-7, DOI:10.1109/CSE.2010.20.

[16] Ivona Brandic, Vincent C. Emeakaroha, Michael Maurer, Schahram Dustdar, Sandor Acs, Attila Kertesz, Gabor Kecskemeti, "LAYSI: A Layered Approach for SLA-Violation Propagation in Self-manageable Cloud Infrastructures", 34th Annual IEEE Computer Software and Applications Conference Workshops, Seoul, 19-23 July 2010, PP:365–370,ISBN:978-1-4244-8089-0,DOI:10.1109/ COMP SACW.2010.70.

[17] Rajkumar Buyya, Saurabh Kumar Garg, Rodrigo N. Calheiros, "SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture, and Solutions", International Conference on Cloud and Service Computing, Hong Kong, 12-14 December 2011, PP: 1 – 10, ISBN: 978-1-4577-1635-5, DOI: 10.1109/CSC.2011.6138522.

[18] K C Gouda, Radhika T V, Akshatha M, "Priority Based Resource Allocation Model for Cloud Computing", International Journal of Science, engineering and

Technology Research, Volume 2, Issue 1, January 2013, PP: 215-219, ISSN: 2278 – 7798 .

[19] Ahmed Amamou, Manel Bourguiba, Kamel Haddadou ,Guy Pujolle, "A Dynamic Bandwidth Allocator for Virtual Machines in a Cloud Environment", 9th Annual IEEE Consumer Communications and Networking Conference- Multimedia & Entertainment Networking and Services, Las Vegas, 14-17 January, 2012, PP: 99 – 104, ISBN:978-1-4577-2070-3, DOI: 10.1109/CCNC.2012.6181065.

[20] Rajkumar Buyya, Rajiv Ranjan, Rodrigo N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services", Algorithms and architectures for parallel processing. Springer Berlin Heidelberg, 2010. PP: 13-31.

[21] Rajeshwari B S, Dakshayini M, "Optimized Service Level Agreement Based Workload Balancing Strategy for Cloud Environment," Advance Computing Conference (IACC), 2015 IEEE International, PP: 160-165, 12-13 June 2015, Print ISBN: 978-1-4799-8046-8, DOI: 10.1109/IADCC. 2015.7154690.

[22] https://en.wikipedia.org/wiki/Service-level_agreement.

[23] John W. Rittinghouse, James F. Ransome, "Cloud Computing: Implementation, management, and security", crc press, 2009.

[24] Demystifying_The_Cloud_eBook[1].pdf.