



## International Journal of Engineering, Business and Enterprise Applications (IJEBA)

[www.iasir.net](http://www.iasir.net)

### Implementation of Agent Based System for Monitoring and Controlling SLA in Cloud Environment

Sarwan Singh<sup>1</sup>, Manish Arora<sup>2</sup>

<sup>1</sup>Research Scholar, Sai Nath University, Jharkhand, India

<sup>2</sup>Joint Director (Systems), National Institute of Electronics and Information Technology, Chandigarh, India.

**Abstract:** Cloud computing is basically a utility based model where Cloud user pays in per usage mode for services being hired. The services are provided on mutually agreed terms and conditions called Service Level Agreement (SLA). The various SLA parameters being agreed are to be monitored while Cloud services are being used and violation if any has to be highlighted. Software Agent has being designed to Monitor and Control SLA in Cloud environment. This paper presents the implementation of Monitoring and Controlling SLA Agent (MACSLA) which monitors and generates alerts in case of any violation of resource allocation. This proposed paper will help both Cloud service provider and Cloud user in controlling agreed levels of SLA parameters.

**Keywords:** Agent, Cloud Computing, Service Level Agreements (SLA), CloudSim, Monitoring & Controlling SLA Agent

#### I. Introduction

Cloud computing is an on-demand computing where cloud users are provided with pool of Hardware and/or Software resources. These resources are used on pay-per-use basis [1]. Cloud service provider provides services like IaaS, PaaS and SaaS to cloud user based on terms and conditions agreed upon called Service Level Agreement (SLA)[2]. Cloud provider faces many challenges in providing reliable services as per SLA at both implementation and deployment stages (Hardware and Software). SLA is considered as the most important element that provides some degree of assurance to both Cloud users and Cloud providers in the Cloud paradigm. The SLA defines the scope of usage and provision of resources, thus plays a vital role in enforcing Quality of Service (QoS).

The SLA defines various resources like RAM, bandwidth, MIPS etc. and their desired level. The virtual machine (VM) is configured according to these resources and their level for providing to Cloud user. Any deviation from these levels may contribute to overheads to both cloud user and Cloud provider. During run time the SLA as agreed and actual being provided can be monitored using software based on agent technologies. Software agents are capable of making run time decision. This motivated to design and implement a dynamic and volatile agent based system for monitoring the terms mentioned in SLA and act timely. This paper presents the implementation of agent based system for monitoring and controlling SLA using CloudSim, JADE with support of MySQL as database. The paper has been organized into four sections; section II reviews the background of work, section III briefs the implementation approach, section IV gives the conclusion and scope for future work.

#### II. Background of Work

Service Level Agreement (SLA) is a document that includes a description of agreed services, service level parameters, guarantees, actions and remedies for all violations cases. Thus SLA is an important agreement of negotiations between Cloud service provider and Cloud user. The SLA also helps in determining the resource/services availability, reliability, scalability, security, etc. The Cloud user logs in to access the Virtual Machine (VM) having the agreed SLA resources. The SLA resources being assigned during runtime are to be monitored for any violations. Software agents can play a vital role in monitoring such violations in resource allocation while providing services to Cloud users.

The agents are considered to be autonomous entities, such as software programs or robots [3]. The agents can share a common interest or they can pursue their own interests. Agent must be able to change its behaviour based on changes occurring in its environment [4]. Agent should be reactive, autonomic, collaborative in behaviour, adaptive, etc. Agents are implemented using Java Agent Development Framework (JADE).

Based on the review of related research, various SLA parameters currently used in ensuring Quality of Service (QoS) in Cloud environment, an agent based Monitoring And Controlling SLA (MACSLA) has been modelled and designed [5].

The MACSLA is designed to work in three layers, the Cloud Administrator/Monitor will interact with MCA in Presentation Layer and Business Logic layer has Term Collector (TC) and Term Monitor (TM) which interacts with Database Layer beneath.

The implementation of design MACSLA is done using simulation. Simulation testing before actual implementation reduces lot of risk related with actual deployment. Various Cloud simulation tools are available in the market, CloudSim, CloudAnalyst, SPECI, Green-Cloud, OCT (Open Closed Testbed), Open Cirrus, GroudSim, Network CloudSim, EMUSIM, DCSim, iCanCloud, GDCSim, MDCSim, CDOSim, TeachCloud.[6] CloudSim is a Java based simulation tool for cloud environment. It can be used with Eclipse IDE or Netbeans IDE. Java runtime environment is required to run eclipse IDE. Different versions of eclipse IDE are available to run CloudSim such Indigo, Juno, Luna, etc.

### III. Implementation Approach

To implement Monitoring and Controlling SLA (MACSLA), three-layered architecture is used as shown in Figure 1. The Cloud Administrator/monitor interacts with Monitoring and Controlling Agent (MCA) that performs at *Presentation Layer*. Presentation Layer contains Java based Graphical User Interface (GUI) to show any violation in resource allocation to Virtual Machine. MCA is implemented using JADE.

The next layer beneath Presentation Layer is *Business Logic Layer*. Business Logic Layer consists of Term Collector (TC) and Term Monitor (TM). Term Collector (TC) gathers the SLA parameters from Cloud service provider whenever new cloud user is registered. The Term Monitor (TM) gathers information from Cloud environment from time to time to maintain QoS being provided by the Cloud service provider. The information collected as TC i.e. SLA's agreed upon and data gathered by TM i.e. the SLA's being provided, are compared and any violations, if found, are projected to Presentation Layer. At lowest level, *Database Layer* is present which store the data in the MySQL Relational Database Management Systems (RDBMS).

The Presentation Layer and Business Logic Layer are implemented in CloudSim using Eclipse Indigo IDE. CloudSim jar files are imported in Eclipse project and few more class files are added to implement Terms Collector (TC), Term Monitor (TM) activities along with database connectivity. JADE package is imported to implement Monitoring And Controlling (MAC) Agent [7]. MAC Agent will gather running SLA parameters from Term Monitor(TM), and compare these with agreed SLA parameters received from Term Collector (TC) and generate runtime Alerts to Cloud users/Administrators.

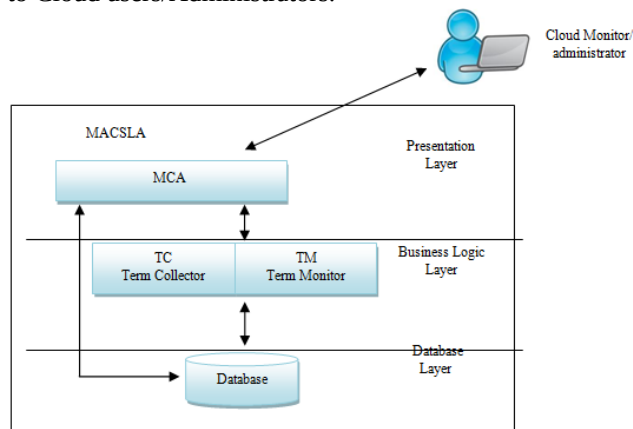


Figure 1: Design of Monitoring and Controlling SLA (MACSLA)

#### A. IMPLEMENTATING DATABASE LAYER

The designed database is implemented in open source MySQL Relational Database Management System (RDBMS). Eight tables have been identified to be used in CloudSim simulation and shown in Table 1.

Table 1: Tables in CloudSim database

| Table name         | Description                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------|
| VmMaster           | Virtual Machine Master table has details e.g. virtual Machine ID, virtual Machine name, etc.               |
| HostResource       | Host Resource Master table has details of Host configuration e.g. RAM, MIPS, Bandwidth, core, storage, etc |
| ResourceMaster     | Table contains master data of different type of resources along with their unit of measurement             |
| ResourceRequired   | Table contains resource required by the Virtual Machine as per Agreed SLA e.g. RAM, MIPS, Bandwidth, etc.  |
| ResourceAllocated  | Table contains resources being allocated to the Virtual Machine at runtime.                                |
| SimulationActivity | This table contains logs of entire operation being simulated in CloudSim                                   |

The tables, their attributes and relationship among them are shown in Figure 2 as Schema Diagram. The fields of various tables as shown with different legends, yellow color legend show primary key, red color legend show foreign key attribute.

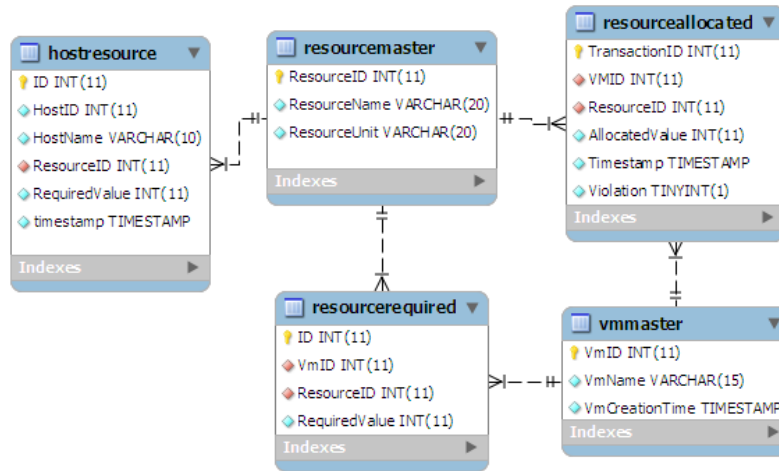


**Figure 2: Schema Design**

These tables are created using Data Definition Language (DDL) statements available in MySQL RDBMS. To illustrate one DDL statement to create 'ResourceAllocated' table is shown in Table 2.

**Table 2: DDL command to create table in MySQL database**

```
CREATE TABLE IF NOT EXISTS 'ResourceAllocated' (
  'TransactionID' INT(11) NOT NULL AUTO_INCREMENT,
  'VMID' INT(11) NOT NULL,
  'ResourceID' INT(11) NOT NULL,
  'AllocatedValue' INT(11) NOT NULL,
  'Timestamp' TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  'Violation' TINYINT(1) NOT NULL,
  PRIMARY KEY ('TransactionID'),
  INDEX 'ResourceID' ('ResourceID' ASC),
  INDEX 'VMID' ('VMID' ASC),
  CONSTRAINT 'ResourceID'
    FOREIGN KEY ('ResourceID')
    REFERENCES 'ResourceMaster' ('ResourceID')
  CONSTRAINT 'VMID'
    FOREIGN KEY ('VMID')
    REFERENCES 'VmMaster' ('VmID')
```

## B. IMPLEMENTING BUSINESS LOGIC LAYER

The Business Logic Layer has two components Term Collector (TC) and Term Monitor(TM). The TC performs various task during its execution.

The task under by Term Collector (TC) are:

- Prepare database, remove previous data from all tables
- Initialize CloudSim
- Create Broker
- Create Virtual Machines
- Create Cloudlets
- Assign Cloudlets to Virtual Machines
- Create Host

Term Collector (TC) will insert the agreed SLA parameters for Virtual Machine (VM) in table 'VmMaster', 'ResourceMaster' table has different type of resource names along with their unit of measurement e.g. RAM (in MB), 'ResourceRequired' table has agreed resources for Virtual Machines in VmMaster table. Table 3 show the java code to create Virtual Machine and inserting Resource's required in the table 'ResourceRequired'. This table has foreign key relationship with 'VmMaster' table and 'ResourceMaster' table.

**Table 3: Java code to implement Term Collector (TC)**

```

//calling method to create VM. parameter : brokerId, Count of VM to be created,
type of VM as defined in Constants class
myCreateVm(brokerId,1, 3) ; myCreateVm(brokerId,1, 2) ;
...
public static void myCreateVm (int brokerId, int vmsCountToBeCreated, int vmType )
{
    ...
    for (int i=0;i<vmsCountToBeCreated; i++){
        vmList.add(new PowerVm(
            vmList.size()+1,
            brokerId,
            Constants.VM_MIPS[vmType],
            Constants.VM_PES[vmType],
            Constants.VM_RAM[vmType],
            Constants.VM_BW,
            Constants.VM_SIZE,
            ...));
        // insert resource required in database
        MyDatabase db = new MyDatabase();
        db.insertResourceRequired(
            vmList.size() ,
            Constants.VM_RAM[vmType],
            Constants.VM_MIPS[vmType] ,
            Constants.VM_SIZE, Constants.VM_BW,
            Constants.VM_PES[vmType]);
        db.closeDB(); db = null;
    }
    ...}

```

Table 4 depicts the java code to assign the cloudlets to Virtual machine, setCloudlettoVM method receives the *cloudletList*, starting-ending offset of the cloudlet in that *cloudletList* along with virtual machine unique identifier (VmID).

**Table 4: Java code to assign Cloudlets to Virtual Machine**

```

setCloudlettoVM(cloudletList, 1, 1, 7);
setCloudlettoVM(cloudletList, 3, 8, 12);
setCloudlettoVM(cloudletList, 2, 13, cloudletList.size() );
...
void setCloudlettoVM(List<Cloudlet> cloudletList, int vmId,
int startCloudletID , int endCloudletID )
{
    for (int i = startCloudletID -1; i< endCloudletID ; i++){
        cloudletList.get(i).setVmId(vmId); ;
    }
}

```

Table 5 code snippet depicts the Host creation method and how the host parameters are inserted in the database.

**Table 5: Java code to create Host**

```

// creating host list with 2 different type of host in round robin fashion
//HOST_MIPS={ 1860,2660} ; HOST_PES={2,2}; HOST_RAM={4096,4096 } ;
    hostList = Helper.createHostList( NUMBER_OF_HOSTS );
...
public static List<PowerHost> createHostList(int hostsNumber) {
    List<PowerHost> hostList = new ArrayList<PowerHost>();
    ...
    hostList.add(new PowerHostUtilizationHistory( i+1,
        new RamProvisionerSimple(Constants.HOST_RAM[hostType]),
        new BwProvisionerSimple(Constants.HOST_BW),
        Constants.HOST_STORAGE, peList,
        new VmSchedulerTimeSharedOverSubscription(peList),
        Constants.HOST_POWER[hostType]));
    // insert Host details in database
    MyDatabase db = new MyDatabase();
    db.insertHostResource((i+1), "Host#"+(i+1),
        Constants.HOST_RAM[hostType], Constants.HOST_MIPS[hostType],
        Constants.HOST_STORAGE,
        Constants.HOST_BW,
        Constants.HOST_PES[hostType]);
    db.closeDB(); db = null;
    ...
    } return hostList;
}

```

The second component of Business Logic Layer is Term Monitor(TM) which monitors the resources being allocated to the Virtual Machine. Basic task of TM comes in picture when VM is being updated while handling Cloudlets. The java code for TM is integrated right at this place. Every time Cloudlet being handled inside VM is updated for any resource, this VM action is saved in the database. Cloud Monitor or MCA agent is accessing the database in order to update the Java based User Interface.

### C. IMPLEMENTATING PRESENTATION LAYER

In Presentation Layer, MCA Agent is implemented using JADE, Table 6 show the Java code for implementing MCA agent. In JADE, to implement Agent java class should extend Agent class available in JADE package (jade.core.Agent). setup () method is overridden to implement user defined task for agent.

**Table 6: Java code to implement MCA agent in JADE**

```
// import statements
public class javaUI extends Agent {
    ... establishing connection to CloudSim database
    static String sql;
    static JTable tableData = null;
    JPanel[][] panelHolder1 = new JPanel[6][4];
    int vRow=6, vCol=4;
    // create GUI interface to monitor SLA for VM
    public JPanel getPanel1 (int vmid)throws Exception{ ... }

    // agent setup method
    protected void setup() { ...
    // run for loop with delay of 1 sec to fetch the current status of SLA parameter from
    database
        for (int i=0;i<3000 ;i++) { ...
            rs=st.executeQuery(" SELECT * FROM resourceallocated where vmid=1 " +
                " and resourceid=1 order by timestamp desc limit 1 " );
            if (rs.next()){
                ...
                if (rs.getInt("violation") ==1){
                    obj.panelHolder1[rec][3].add(new JLabel("true"));
                    obj.panelHolder1[rec][3].setBackground(Color.RED);
                }else{
                    obj.panelHolder1[rec][3].add(new JLabel("false"));
                    obj.panelHolder1[rec][3].setBackground(Color.WHITE);
                }
            }
            try { Thread.sleep(1000);} catch(Exception e){ ... }
        }
    } // setup method ends
} // Agent code ends
```

### IV. Conclusion and Scope for Future Work

Implementation of three layered design of Monitoring And Controlling SLA (MACSLA) using Presentation layer, Business Logic layer and Database layer when deployed in real time scenario helps to monitor the agreed SLA parameters during runtime. The tools of JADE, CloudSim, Java and MySQL database in an integrated way helped the implementation of MACSLA in a trouble free manner. Scope for future enhancement includes verification of the implemented MACSLA

### References

- [1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandie, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing As the 5th Utility", Future Generation Computer System, Volume 25 Issue 6, pp. 599-616, June, 2009
- [2] Mohammed Alhamad, Tharam Dillon, Elizabeth Chang, "Conceptual SLA Framework for Cloud Computing", pp, 606-610, IEEE 2010.
- [3] Sarwan Singh, Manish Arora , "Monitoring and Controlling Multi Level SLA in Cloud Environment using Agent" IJARCSSE, volume 3 issue 7, pp. 1006-1010, July 2013
- [4] Manish Arora, M. Syamala Devi, "Distributed algorithm for multi-agent environment" , IJITKM, volume 4 no.2, pp. 519-525, 2011
- [5] Sarwan Singh, Manish Arora , "Design of Agent Based System for Monitoring and Controlling SLA in Cloud Environment", IJATES, Volume No 03, Special Issue No. 01, March 2015
- [6] Utkal Sinha, Mayank Shekhar "Comparison of Various Cloud Simulation tools available in Cloud Computing" IJARCCCE, Vol. 4, Issue 3, March 2015
- [7] Manish Arora, M. Syamala Devi, "Implementation of Web Based Multi Agent Resource Allocation System" , International Journal of Information Sciences and Application(IJITKM) , ISSN 0974-2255, Volume 2, number 2, 2010