



# Artificial Automation Based Resource Allocation for Deep learning Simulation in a Multi-tenant Cloud

Ramya P<sup>1</sup>, Rohini M<sup>2</sup>

<sup>1</sup>Student, The kavery Engineering College, M.Kalipatti, Tamil Nadu

<sup>2</sup>Assistant Professor, The kavery Engineering College, M.Kalipatti, Tamil Nadu

## ABSTRACT

Multi-tenant technology is one of key competencies for Software as a Service (SaaS) to achieve higher profit and performance for the cloud users. Multi-tenancy enables to share resources of a single database server among multiple tenants. This *multi-tenancy* enables cost reduction for the cloud service user. Database as a Service (DBaaS) is one of the service delivery model in which resources (Memory, CPU and I/O) are accessed remotely by users. The performance of database server depends on the key features such as CPU, I/O and memory for cloud users. In the case of fixed users, static database allocation technique is practiced. But when the number of users demand increase for a specific resource utilization, there occurs performance degradation. The existing static technique will not be sufficient enough when the usage of the resources varies from time to time. It degrades the usage level and performance at DBaaS. To overcome this issue, we propose a Resource Allocation Framework with Genetic Algorithm. It will improve the DBaaS utilization better and helps to increase the performance at the multi-tenant environment.

## 1 INTRODUCTION

### 1.1 CLOUD COMPUTING -THE BASICS

Cloud computing is Internet "cloud" based development and use of computer technology. It is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. Users need not have knowledge of, expertise in, or control over the technology infrastructure "in the cloud" that supports them.

The concept incorporates infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) as well as Web 2.0 and other recent technology trends which have the common theme of reliance on the Internet for satisfying the computing needs of the users.

### 1.2 CHARACTERISTICS OF CLOUD

As customers generally do not own the infrastructure, they merely access or rent, they can avoid capital expenditure and consume resources as a service, paying instead for what they use.

**Cost** is greatly reduced and capital expenditure is converted to operational expenditure. This lowers barriers to entry, as infrastructure is typically provided by a third-party and does not need to be purchased for one-time or infrequent intensive computing tasks. Pricing on a utility computing basis is fine-grained with usage-based options and minimal or no IT skills are required for implementation.

**Device and location independence** enable users to access systems using a web browser regardless of their location or what device they are using, e.g., PC, mobile. As infrastructure is off-site (typically provided by a third-party) and accessed via the Internet the users can connect from anywhere.

**Multi-tenancy** enables sharing of resources and costs among a large pool of users, allowing for:

- a) **Centralization** of infrastructure in areas with lower costs (such as real estate, electricity, etc.)
- b) **Peak-load capacity** increases (users need not engineer for highest possible load-levels)
- c) **Utilization and efficiency** improvements for systems that are often only 10-10% utilized.
- d) **Reliability** improves through the use of multiple redundant sites, which makes it suitable for business continuity and disaster recovery. Nonetheless, most major cloud computing services have suffered outages and IT and business managers are able to do little when they are affected.

- e) **Scalability** via dynamic ("on-demand") provisioning of resources on a fine-grained, self-service basis near real-time, without users having to engineer for peak loads. Performance is monitored and consistent and loosely-coupled architectures are constructed using web services as the system interface.

**Security** typically improves due to centralization of data, increased security-focused resources, etc., but raises concerns about loss of control over certain sensitive data. Security is often as good as or better than traditional systems, in part because providers are able to devote resources to solving security issues that many customers cannot afford. Providers typically log accesses, but accessing the audit logs themselves can be difficult or impossible.

**Sustainability** comes about through improved resource utilization, more efficient systems, and carbon neutrality. Nonetheless, computers and associated infrastructure are major consumers of energy.

## 2. ARCHITECTURE

Cloud architecture, the systems architecture of the software systems involved in the delivery of cloud computing, comprises hardware and software designed by a cloud architect who typically works for a cloud integrator. It typically involves multiple cloud components communicating with each other over application programming interfaces, usually web services.

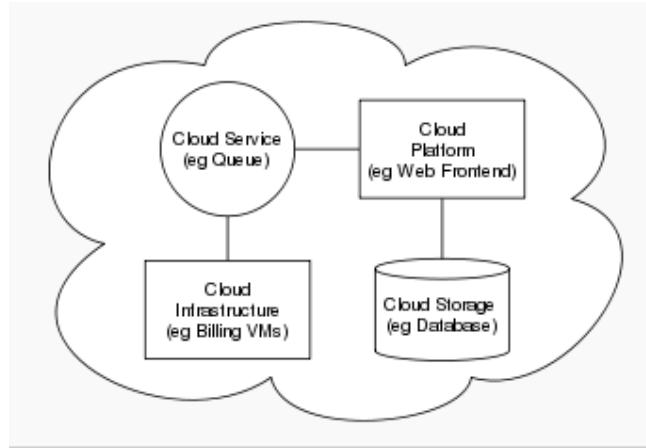


Fig 2 Cloud Computing Sample Architecture

Cloud architecture extends to the client, where web browsers and/or software applications access cloud applications. Cloud storage architecture is loosely coupled, where metadata operations are centralized enabling the data nodes to scale into the hundreds, each independently delivering data to applications or user.

## 3. TYPES

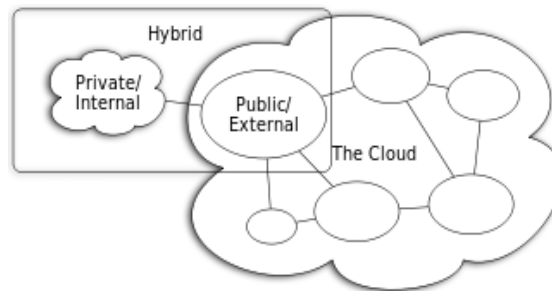


Fig 3 Types of Clouds

### 3.1 Public cloud

Public cloud or external cloud describes cloud computing in the traditional mainstream sense, whereby resources are dynamically provisioned on a fine-grained, self-service basis over the Internet, via web applications/web services, from an off-site third-party provider who shares resources and bills on a fine-grained utility computing basis.

### 3.2 Private cloud

Private cloud and internal cloud are neologisms that some vendors have recently used to describe offerings that emulate cloud computing on private networks. These products claim to "deliver some benefits of cloud computing without the pitfalls", capitalizing on data security, corporate governance, and reliability concerns.

### 3.3 Hybrid cloud

A hybrid cloud environment consisting of multiple internal and/or external providers "will be typical for most enterprises". It is a combination of a private cloud combined with the use of public cloud services where one or several touch points exist between the environments. The goal is to combine services and data from a variety of cloud models to create a unified, automated, and well-managed computing environment.

### 3.4 Community cloud

A community cloud is a [multi-tenant infrastructure](#) that is shared among several organizations from a specific group with common computing concerns. Such concerns might be related to regulatory [compliance](#), such as audit requirements, or may be related to performance requirements, such as hosting applications that require a quick [response time](#).

---

## 4. ROLES

### 4.1 Provider

A cloud computing provider or cloud computing service provider owns and operates live cloud computing systems to deliver service to third parties. The barrier to entry is also significantly higher with capital expenditure required and billing and management creates some overhead. Nonetheless, significant operational efficiency and agility advantages can be realized, even by small organizations, and server consolidation and virtualization rollouts are already well underway.

Amazon.com was the first such provider, modernizing its data centers which, like most computer networks, were using as little as 10% of its capacity at any one time just to leave room for occasional spikes. This allowed small, fast-moving groups to add new features faster and easier, and they went on to open it up to outsiders as Amazon Web Services in on a utility computing basis.

### 4.2 User

A user is a consumer of cloud computing. The privacy of users in cloud computing has become of increasing concern. The rights of users are also an issue, which is being addressed via a community effort to create a bill of rights.

### 4.3 Vendor

A vendor sells products and services that facilitate the delivery, adoption and use of cloud computing. For example:

- Computer hardware (Dell, HP, IBM, Sun Microsystems)
  - a) Storage (Sun Microsystems, EMC, IBM)
  - b) Infrastructure (Cisco Systems)
- Computer software (3tera, Hadoop, IBM, RightScale)
  - a) Operating systems (Solaris, AIX, Linux including Red Hat)
  - b) Platform virtualization (Citrix, Microsoft, VMware, Sun xVM, IBM)

---

## 5. TYPES OF SERVICE

- IaaS Infrastructure as a Service, i.e. systems which offer a fully virtualized architecture based on a service oriented interface, as an example it is possible to create and deliver a (virtual) machine to a final user leaving him fully administrative control.
- PaaS Platform as a Service, i.e. the system offers a virtualized platform, usually with a web interface, on which it is possible to develop applications, customize systems,
- SaaS Software as a Service, i.e. the system hosts applications that can be accessed directly through a web interface (as an example GoogleApps).

---

## 6. MULTI-TENANCY IN SaaS CLOUD COMPUTING ENVIRONMENT

### Definition

Multitenancy refers to a principle in [software architecture](#) where a single instance of the [software](#) runs on a server, serving multiple client organizations (tenants). Multitenancy is contrasted with a multi-instance architecture where separate software instances (or hardware systems) are set up for different client organizations. With a multitenant architecture, a [software application](#) is designed to virtually [partition](#) its data and configuration, and each client organization works with a customized virtual application instance.

In [cloud computing](#), multi-tenancy means that a SaaS (Software as a Service) vendor provides a single version of its software for all its customers. This differs from a single-tenant hosted solution, where the application is housed on a vendor's server but the codebase is unique for each customer.

### 6.1 Benefits of Multi-tenancy

The advantages of a multi-tenancy SaaS over a third-party-hosted, single-tenancy application include the following:

- Lower costs through economies of scale: With a single-tenancy-hosted solution, SaaS vendors must build out their data center to accommodate new customers. In contrast, in a multi-tenant environment, new users get access to the same basic software, so scaling has far fewer infrastructure implications for vendors (depending on the size of the application and the amount of infrastructure required).
- Shared infrastructure leads to lower costs: SaaS allows companies of all sizes to share infrastructure and data center operational costs. Users don't need to add applications and more hardware to their data centers, and some small- to medium-sized businesses don't even need data centers if they utilize SaaS.
- Ongoing maintenance and updates: End users don't need to pay costly maintenance fees in order to keep their software up to date. New features and updates are included with a SaaS subscription and are rolled out by the vendor.
- Configuration can be done while leaving the underlying codebase unchanged: Although on-premises applications and single-tenant-hosted solutions are often customized, this endeavor is costly and requires changes to an application's code. Additionally, this customization can make upgrades time-consuming, because an upgrade might not be compatible with your customization.
- Vendors have a vested interest in making sure everything runs smoothly: Multi-tenant SaaS providers have, in a sense, all their eggs in one basket. Although this sounds dangerous, it's a benefit to end users.

### 6.2 DATABASE as a SERVICE (DBaaS)

"Database as a Service (DBaaS) - Cloud Computing" offers a common shared service to the enterprise as a whole. Database as a Service is a service where clients pay per use and get access to the cloud database service on-demand. This kind of service is much attractive and affordable for clients, since the costs related to the maintenance, hardware and software that runs are bear by the service provider.

### Benefits

Typically, DBaaS provides customers with many of the same benefits they expect from any cloud service:

- Flexible
- Scalable
- On-demand platform
- Lower cost

that's oriented toward self-service and easy management, particularly in terms of provisioning their own environments. Such services also provide enough monitoring capabilities to track performance and usage, be alerted to issues that might arise and generate at least some degree of analytics.

---

## 7. SCOPE OF THE PROJECT

The objective is to develop a model that provides a better and effective resource allocation strategy with in the cloud environment. Multi-tenancy enables to share resources of a single database server among multiple tenants. This multi-tenancy enables cost reduction for the cloud service provider. However, resource sharing can affect a tenant's performance due to resource demands of other tenant's workloads. During the conditions like number of tenants increase's, in the resource utilization and according to their demands.

A multi tenant SaaS product should be efficient enough to scale seamlessly without compromising on:

- Reliability

- Availability
- Performance

When the number of tenants and users grow, the number of IOs which hits the database will also increase which is susceptible for performance degradation. The performance of database server resources depends on the key features such as CPU, I/O and memory, for each tenant. The key challenge of a shared DBMS is static allocation of resources to tenants, while the occurrence of low overheads and scaling to large numbers of tenants.

### 1. Multi-tenancy Security Risks and Countermeasures

Cloud Computing is quickly being adopted by organizations and businesses alike to help increase profit margins by decreasing overall IT costs as well as provide clients with faster implementation of services. The majority of the cloud service providers offer multi-tenancy to capitalize on the associated economies of scale which also translates into savings for the end user. Multi-tenancy has made cloud computing popular by allowing businesses to benefit from reduced costs yet continue to gain access to data and applications within a cloud environment. In the multi-tenancy model, many user's data and resources are located in the same computing cloud, and are controlled and distinguished through the use of tagging for the unique identification of resources owned by individual user.

### 2. An Approach to Build Multi-Tenant SaaS Application with Monitoring and SLA

Multi-tenancy is one of the most important concepts for any SaaS application. Based on it, real cloud computing aims for "better resource utilization" for SaaS providers and "Pay as you go" for consumer. There are many ways to develop SaaS application based on hosting environment, architecture to follow, and available development skills. One can develop SaaS and deploy it using proprietary online framework like Salesforce.com, Google AppEngine, Microsoft Azure etc. SaaS also can be developed using core programming framework like JDK for java and .NET. Here, we have to focus on monitoring, billing, subscription and most important configurability per tenant. Proposed architecture includes monitoring, tenant management, tenant administration, tenant configuration and large data management services.

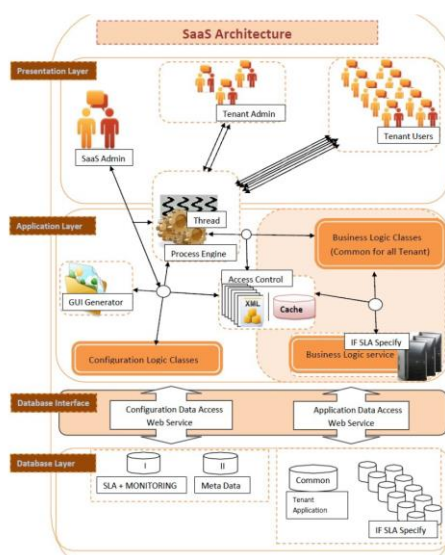


Fig SaaS Architecture

### 3. A Multi-tenant Oriented Performance Monitoring, Detecting and Scheduling Architecture Based on SLA

Generally speaking, web service handles all the requests equally according to the Best-Effort service model, which uses First in First out (FIFO) scheduling strategy. However, the performance improvements realized by IntServ and DifServ will be seriously affected and unable to meet the different requirements of various users. For example, when the load is heavy, it will discard all the overloaded user requests, or prolong response time without consideration of prioritization. Such abuse also exists in multi-tenant oriented services. All SaaS applications share a high-performance server; therefore failure of any SLA control may lead to system instability and the decline of server's overall performance. At present, the research on assurance mechanisms of service quality has become a hotspot.

Research on service quality oriented distribution or scheduling mechanism in the area of grid computing is relatively mature. There's Globus Architecture for Reservation and Allocation (GARA) which is able to provide peer-to-peer grid service based on Quality of Service (QoS). GARA implements a uniform management mechanism for diverse resources, such as CPU, network, storage equipment. It also offers several operations on service quality management, including resource selection, distribution and release, quality of service assurance, and point-to-point resource management. Database-as-a-Service (DBaaS) is gaining significant momentum with more vendors providing offerings. Amazon RDS offers the full capabilities of a familiar MySQL database by running MySQL instances in different EC2 instances (VM Instances) for different tenants. Microsoft SQL Azure provides a SQL server database for different tenants in a shared pool of SQL server instances with a controlled allocation of resources for each database.

#### 4. A Hybrid Approach to Placement of Tenants for Service-based Multi-tenant SaaS Application

In the multi-tenancy application, the number of servers is limited and it restricts the number of Web service instances deployed on servers. To reduce the cost of ownership, one of the important problems is that how to optimally place tenants to the limited number of servers to maximize the total supported number of tenants without violating their Service

Level Agreement (SLA) requirements. To solve the problem, this paper proposes the Tenant Placement Strategy (TPS).

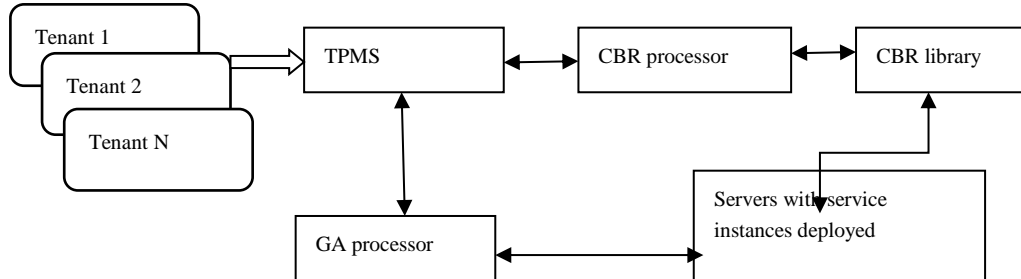


Fig The Framework of Tenant Placement System

The resource estimation model is used to computing the residual resources of servers to prevent the system being overloaded, and the resource types include CPU, memory and DiskIO. In order to provide tenants execution plans with appropriate quality, SLA must be established in the multi-tenant application and it usually describes QoS of a Web service instance such as response time, cost, availability, reliability and throughput and so on.

#### 5. Symphony: A Scheduler for Client-Server Applications on Coprocessor-based Heterogeneous Clusters

The problem of a multi-tenant GPU-based heterogeneous cluster delivering acceptable response times (i.e., a response time that is less than or equal to the pre-specified response time) in the presence of load spikes. Response time is an important part of the system's Quality-of-Service (QoS), and is also the main concern of the client. For a heterogeneous cluster to handle client-server applications with load spikes, a scheduler that enables dynamic sharing of heterogeneous resources is necessary. To propose a novel cluster-level scheduler, *Symphony*, that enables efficient sharing of heterogeneous cluster resources while delivering acceptable client request response times despite load spikes. *Symphony* manages client requests of different applications by assigning each request a priority based on the load and estimated processing time on different processing resources like the CPU. It then directs the highest priority application to issue requests to suitable processing resources within the cluster nodes. Specifically, this paper makes the following contributions:

## 8 SYSTEM ANALYSIS

### 8.1 EXISTING SYSTEM

Cloud computing provides a cost-effective and customizable means through which scalable computing power and diverse services (such as computer hardware and software resources, networks and computing infrastructures), diverse application services, business processes to personal intelligence and collaboration are delivered as services to large-scale global users whenever and wherever they need. As a result most of the business in the large and SMB (Small and Medium Business) categories have already started or planning to start migration to the cloud. The primary derivatives that are being looked at are the cost effectiveness and lack of effort in scaling up and scaling down the infrastructure. Traditionally people have been working on single server applications and to ensure the quality of such applications several techniques and approaches were developed and practiced over time. It is important to figure out the relative balance in the efforts/tools/resources that need to be employed to make sure that each of the factors affecting the end-user experience is adequately taken care of without shooting the price of quality.

A multi tenant SaaS product should be efficient enough to scale seamlessly without compromising on Reliability, Availability and Performance. Large scale applications which are built with the intention of handling thousands of users accessing in a concurrent fashion is to be well equipped and architected to handle a medium sized customer with few hundreds of users to a large customer with fairly huge number of active users. When the number of tenants and users grow, the number of IOs which hits the database will also increase which is susceptible for performance degradation.

The performance of database server resources depends on the key features such as CPU, I/O and memory, for each tenant. The key challenge of a shared DBMS is static allocation of resources to tenants, while the occurrence of low overheads and scaling to large numbers of tenants. The technique should focus to contribute in: designing mechanisms to support the promised resources and proposing low-overhead techniques to increase the performance in multi-tenant environment.

#### Performance depends on:

- CPU
- I/O

- Memory
- Network Bandwidth

### Limitations of the Existing System

- The RDBMS memory storage is mostly used for storing and retrieving of data's to number of tenants. While there are many uses of memory in a relational DBMS, while we allocate static memory to each tenant (a certain amount), it will not be sufficient enough when the tenant need exceeds at a session, hence the usage of the resource, storage varies according to their application.
- In the case of buffer pool, this is used as a cache. The buffer pool is a cache of database pages that is managed using a page replacement strategy (e.g., LRU). The old session pages may be removed by the new pages, when there is no space for newly requested data's. And the query varies according to the tenants, it may lead to overhead.
- Inefficient Dynamic Resource allocation framework for memory scheduling process among the users.

## 8.2 PROPOSED SYSTEM

In the proposed system, a Resource allocation strategy is used to enhance the performance in DBaaS as a Service among users. The performance degradation in SaaS service occurs during the multi-tenancy. In the case of, hardware resource allocated to a tenant (e.g. an organization) the performance should be maintained according to the SLA. In some case, during the abnormal conditions like overload or bursty condition (i.e.) more number of users is in need of hardware resource at same time of a tenant. The provider allocated memory resource to the tenant should be efficiently scheduled to the "N" number of users, even in the overload condition also.

To achieve the better performance in multi-tenancy within a tenant. We use the methods:

- Monitor
- Analyser
- Predictor
- Allocator

### 8.2.1 Advantages of proposed system

- The memory utilization is better and faster for the end user and the tenant can effectively allocate the resource by using the resource allocation framework.
- Achieve higher resource utilization if there is a higher variance in user's number among the tenant up to the SLA assigned DBaaS service by the cloud service provider.
- Enhance the performance and, it is effective and efficient.
- Ease of use with flexibility.

## 8.3 SOFTWARE DESCRIPTION

### 8.3.1 Programming Language: java

Java is a high-level, third generation programming language, like C, Fortran, Smalltalk, Perl, and many others. We can use Java to write computer applications that crunch numbers, process words, play games, store data or do any of the thousands of other things computer software can do. Compared to other programming languages, Java is most similar to C. However although Java shares much of C's syntax, it is not C.

Java helps to contribute to Java's robustness and provides a mechanism whereby the Java environment can ensure that a malicious applet doesn't steal all of the host's CPU cycles. fortunately multithreading is so tightly integrated with Java, that it makes Java rather difficult to port to architectures like Windows 3.1 or the PowerMac that don't natively support preemptive multi-threading. The exact algorithm used for garbage collection varies from one virtual machine to the next.

### Java platform

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any supported hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java byte code, instead of directly to platform-specific [machine code](#). Java byte code instructions are analogous to machine code, but are intended to be [interpreted](#) by a [virtual machine](#) (VM) written specifically for the host hardware. [End-users](#) commonly use a Java Runtime Environment (JRE) installed on their own

machine for standalone Java applications, or in a Web browser for Java applets. Standardized libraries provide a generic way to access host-specific features such as graphics, [threading](#), and networking.

### 8.3.2 Netbeans

The NetBeans IDE is an award-winning Integrated Development Environment available for Windows, Mac, Linux, and Solaris. The NetBeans project consists of an open-source IDE and an application platform which enable developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as PHP, JavaScript and Ajax, Ruby and Ruby on Rails, Groovy, and C/C++. NetBeans IDE is an [open-source](#) integrated development environment. NetBeans IDE supports development of all Java application types out of the box. Among other features are an [Ant](#)-based project system, Maven support, [refactoring](#), and control.

### 8.3.3 PostgreSQL

PostgreSQL, often simply Postgres, is an [object-relational database management system](#) (ORDBMS) available for many platforms including [Linux](#), [FreeBSD](#), [Solaris](#), [Microsoft Windows](#) and [Mac OS X](#). It is released under the PostgreSQL License, which is an [MIT-style license](#), and is thus [free and open source software](#). PostgreSQL is developed by the PostgreSQL Global Development Group, consisting of a handful of volunteers employed and supervised by companies such as [Red Hat](#) and [EnterpriseDB](#)

The vast majority of Linux distributions have PostgreSQL available in supplied packages. Mac OS X, starting with [Lion](#), has PostgreSQL server as its standard default database in the server edition, and PostgreSQL client tools in the desktop edition. PostgreSQL has built-in support for 3 procedural languages:

- Plain SQL (safe). Simpler SQL functions can get [expanded inline](#) into the calling (SQL) query, which saves function call overhead and allows the query optimizer to "see inside" the function.
- PL/pgSQL (safe), which resembles Oracle's [PL/SQL](#) procedural language.
- [C](#) (unsafe), which allows loading custom [shared libraries](#) into the database. Functions written in C offer the best performance, but bugs in code can crash and potentially corrupt the database. Most built-in functions are written in C.

### 8.3.4 Cloudsim

CloudSim is proposed as a framework for modelling and simulation of Cloud Computing environments to support performance evaluation of policies for resource provisioning / application scheduling / policies of federation of Clouds (in a repeatable and controllable manner).

The CloudSim is implemented at the next level by programmatically extending the core functionalities exposed by the Grid Sim layer. CloudSim provides novel support for modeling and simulation of virtualized Cloud based data center environments such as dedicated management interfaces for VMs, memory, storage, and bandwidth. CloudSim layer manages the instantiation and execution of core entities (VMs, hosts, data centers, application) during the simulation period.

#### Virtual Machine

This class models an instance of a VM, whose management during its life cycle is the responsibility of the Host component. As discussed earlier, a host can simultaneously instantiate multiple VMs and allocate cores based on predefined processor sharing policies (space-shared, time-shared). Every VM component has access to a component that stores the characteristics related to a VM, such as memory, processor, storage, and the VM's internal scheduling policy, which is extended from the abstract component called VMScheduling.



## 8.4 SYSTEM DESIGN

### 8.4.1 CONTEXT DIAGRAM

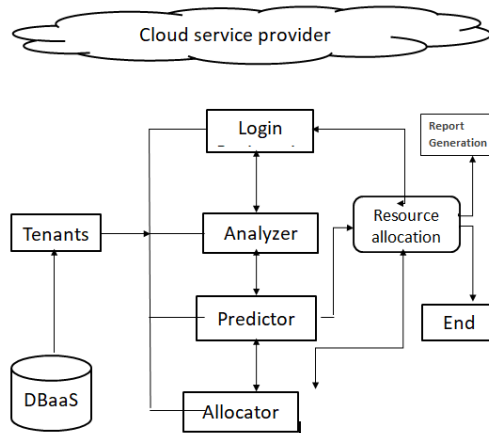


Fig Context Diagram

### 8.5 MODULE DESCRIPTION

The modules are

- Login & Registration
- Analyzer
- Predictor
- Allocator
- Report Generation

#### 8.5.1 LOGIN & REGISTRATION:

First the user's register their request to the tenant through the Request Registration form. The tenant can able to view the list of user's and their requirements. Only the tenant can submit the registered request from user's to cloud environment to utilize the resource as service along with the authentication.

Tenant has an authorized entry into the cloud environment, they are provided with a username and password to login and do the process. The Parameters are User ID, Service type-regarding their application are used to analyze the resource they are needed.

#### 8.5.2 ANALYZER:

Analyzer examines monitored data, and provides input to the Predictor and Allocator. Two jobs are performed by the Analyzer:

Counting the number of current usage memory and resource information

- Comparing the observed workloads with the assigned storage in terms of SLA

The various parameters that used in the analyzer module include are CPU usage, Memory usage, and Network usage. But currently we focus on first two parameters. With the help of SLA and user request form information, can able to calculate the resource required. In view user's registration form we can able to identify the number of user's and from that we can give the input to analyzer.

#### 8.5.3 PREDICTOR:

The workload prediction has two purposes:

- To decide on the demand pattern of a workload
- To make some model report for the predicated memory

Here, the inbuilt process takes progress in backend and updated, hence it takes some time for processing. At the time of cloud environment initialization it starts to predict the current cluster usage, average CPU usage per day, available resource and requested resource , average up/down time per day etc..

#### 8.5.4 ALLOCATOR:

A tenant can consume resources according to the SLA. If the real demands of a tenant are more, then it cannot be observed. Likewise, the resource is allocated to the each user's. Based on the results from the predictor it shows the available clusters to the user requirement.

After the conformation from the tenant the resources are allocated in the basis of Genetic algorithm by using the allocation framework. If the available memory is greater than or equal to the user requested memory, it proceeds the allocation else it waits till the cluster get free space for utilization, in that unavailable case that process is indicated as killed and proceeds.

#### 8.5.5 REPORT GENERATION:

After completion of resource allocation for the requested user from clusters present in cloud. The report is generated with the help of information from the analyzer, predictor and allocator. It shows the reports like average utilization per cluster, cluster usage per hour, waiting and running jobs (i.e.) the request for resource by user, average system utilization.

## 9. SYSTEM TESTING

Testing is the one step in the Software Engineering process that could be viewed as destructive rather than constructive. Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing is representing an interesting anomaly for the software.

### 9.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of the software design. This project comprises the set performed by an individual programmer prior to the integration of the unit into a larger system. In this project the unit testing is done over the module login.

### 9.2 INTEGRATION TESTING

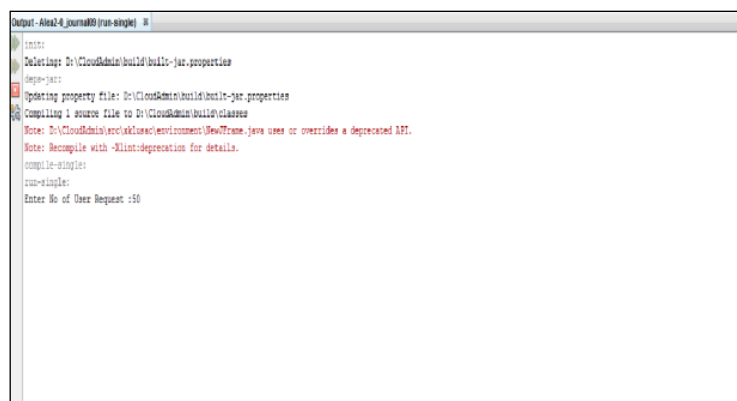
The data can be lost across an interface; one module can have adverse effort on another, sub function when combined may not produced the desired function. Integration testing is a systematic technique for constructing the program while at the same time conducting test to uncover errors associated within the interface.

The objective is to take unit-tested module and built the program structure that has been dictated by design. In this project integration testing is done at the predictor phase and Report Generation.

The predictor module collects all data which is necessary for the process to check the availability of the memory and allocate. The data are gathered from the module such as Analyzer and Allocator.

The resource parameters such as RAM, CPU and Memory are calculated and the data are updated at the server. It helps for better performance and to know the availability.

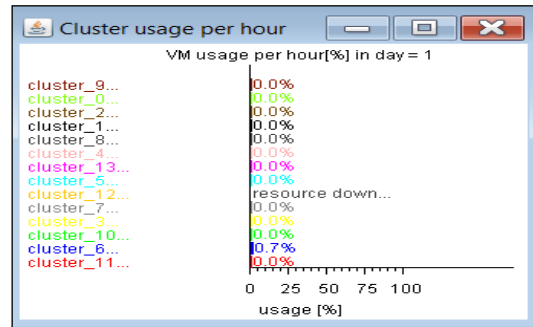
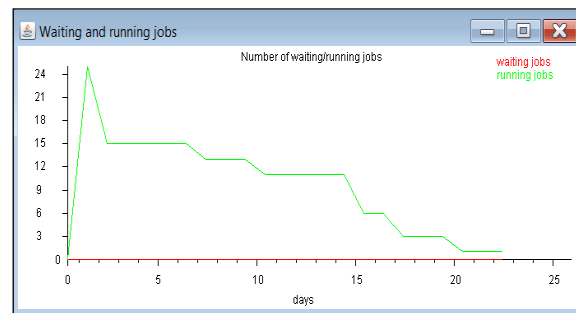
#### Run simulation:



```

Output: Ahs2-4_jurnal@run-single: ~
rm -f build\build-jar.properties
rm -f build-jar.jar
rm -f build\classes
rm -f build-jar.jar
Note: Recompile with -Xlint:deprecation for details.
compile-single:
run-single:
Enter No of User Request :50

```

**Cluster Usage:****Graphical View:****CONCLUSION & FUTURE ENHANCEMENT**

We proposed a resource allocation framework with genetic algorithm for DBaaS. In the framework, resources are allocated with the objective of high resource utilization under SLA violation and fairness constraints. The memory utilization is better and faster for the end user and the tenant can effectively allocate the resource by using the resource allocation framework. The proposed framework over DBaaS can achieve higher resource utilization when there is a higher variance in user's number among the tenant up to the SLA assigned DBaaS service by the cloud service provider. The proposed method for hardware resource allocation over end users among a particular tenant will enhance the performance and, it is effective.

In future, we have an idea of applying effective monitoring algorithms to further improve the performance of DBaaS allocation. May concentrate on the individual user priority based on their hit ratio and make the service avail as the SLA favour to tenants.

**REFERENCES**

1. Wonjae Lee and Min Choi, "Multitenancy – Security Risks and Countermeasures", Fifth International Conference on Cloud Computing, IEEE computer society, 2012, pp. 970-971.
2. Piyush Aghera, Sanjay Chaudhary and Vikas Kumar, "An Approach to Build Multi-Tenant SaaS Application with Monitoring and SLA", International Conference on Communication Systems and Network Technologies, IEEE computer society, 2012, pp. 658-66.
3. Xu Cheng, Yuliang Shi, and Qingzhong Li, "A Multitenant Oriented Performance Monitoring, Detecting and Scheduling Architecture Based on SLA", IEEE-2009, pp.599-604.
4. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and et al., "Above the clouds: A Berkeley view of cloud computing," Computing, no. UCB/EECS-2009-28, 2009. [Online]. Available: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>
5. G. Liu, "Research on independent saas platform," in Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on, April 2010, pp. 110–113.
6. C. D. Weissman and S. Bobrowski, "The design of the force.com multitenant internet application development platform," in Proceedings of the 35th SIGMOD international conference on Management of data, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 889–896. [Online]. Available: <http://doi.acm.org/10.1145/1559845.1559942>.

- 
7. Foster I. and Kesselman C. "Globus: A Toolkit-Based Grid Architecture". Foster, I. and Kesselman, C. (eds.), *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999. pp. 259-278.
  8. Amazon Rational Database Service <http://aws.amazon.com/rds/>
  9. SQL Azure: Database-as-a-Service <http://www.microsoft.com/windowsazure/sqlazure/>
  10. A.Stefan, G.Torsten, J.Dean, K.Alfons, R.Jan, "Multi-tenant databases for software as a service: schema-mapping techniques," Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1195-1206.