DHRUV GARG 16 BCE 1190

ANALYSIS AND COMPARISON OF 3 HPC PARADIGMS

DIGITAL ASSIGNMENT 1

INTRODUCTION

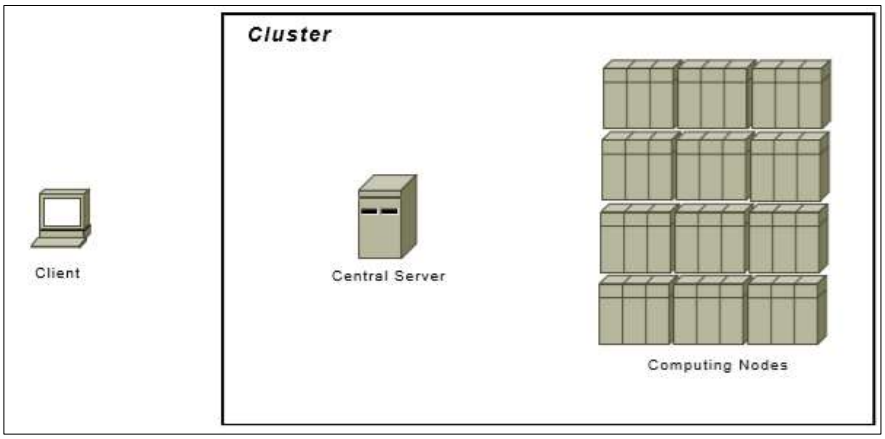
The needs and expectations of modern-day applications are changing in the sense that they not only need computing resources (be they processing power, memory or disk space), but also the ability to remain available to service user requests almost constantly 24 hours a day and 365 days a year. These needs and expectations of today's applications result in challenging research and development efforts in both the areas of computer hardware and software.

It seems that as applications evolve they inevitably consume more and more computing resources. To some extent we can overcome these limitations. For example, we can create faster processors and install larger memories. But future improvements are constrained by a number of factors, including physical ones, such as the speed of light and the constraints imposed by various thermodynamic laws, as well as financial ones, such as the huge investment needed to fabricate new processors and integrated circuits.

The obvious solution to overcoming these problems is to connect multiple processors and systems together and coordinate their efforts. The resulting systems are popularly known as parallel computers and they allow the sharing of a computational task among multiple processors.

CLUSTER COMPUTING: Description and applications

A cluster is a type of parallel or distributed computer system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource. The various nodes involved in cluster are normally connected to each other using some fast local area networks.



Some definitions revisited:

- **Process** is a program that is running into memory. A **thread** (also known as a light process) is a program that shares an amount of memory with another thread belonging to the same **parent process**.
- Multiple processes can be grouped into **jobs**.
- A job may be a **single process**, a **group of processes** or even a **batch process**. In HPC clusters, a job is a set of processes that have been automatically launched by the **scheduler** by way of a user's submitted script. A job may result in N **instances** of the same program on N nodes or processors of a cluster.
- These jobs run on **nodes**. Each node has a set of "p" CPUs, an amount of memory, one or several network interfaces and have a storage unit (local disk).
- Several nodes are connected to a **computing network** generally a low latency network (Myrinet, Infiniband, Numalink,etc.).
- **Jobs** maybe parallel or sequential.
    - **Sequential job**: is a unique process that runs on a unique processor of a unique host.
    - **Parallel job**: Several processes or threads that can communicate via a specific library (MPI, OpenMP, threads). Some of them are 'shared memory' parallel jobs (they run on a unique multiprocessor

host) and some are 'distributed memory' parallel jobs (they may run on several hosts that communicate via a low latency high speed network)

- Finally, jobs can come in the form of a **Batch script** or in the **interactive type**. A batch job is a script. A shell script is a given list of shell commands to be executed in a given order. An interactive job is usually an allocation upon one or more nodes where the user gets a shell on one of the cluster nodes.

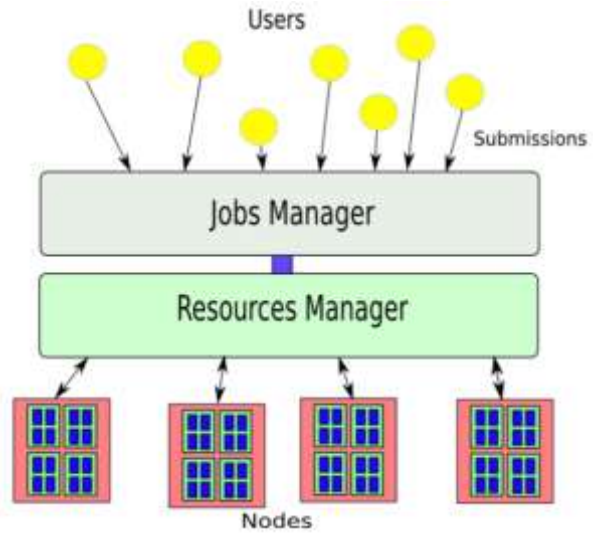
Resource and Job Management system (RJMS) or Batch Scheduler

It is a particular software of the cluster that is responsible to distribute computing power to user jobs within a parallel computing infrastructure.

The goal of a Resource and Job Management System is to satisfy users demands for computation and achieve a good performance in overall system's utilization by efficiently assigning jobs to resources.

This work involves three principal abstraction layers:

1. The **declaration of a job** where the demand of resources and job characteristics take place
2. The **scheduling of the jobs** upon the resources
3. The **launching and placement of job instances** upon the computation resources along with the job's control of execution.



In this sense, the work of a RJMS can be decomposed into three main subsystems:

Job Management >> **Scheduling** >> **Resource Management**.

Resource matching

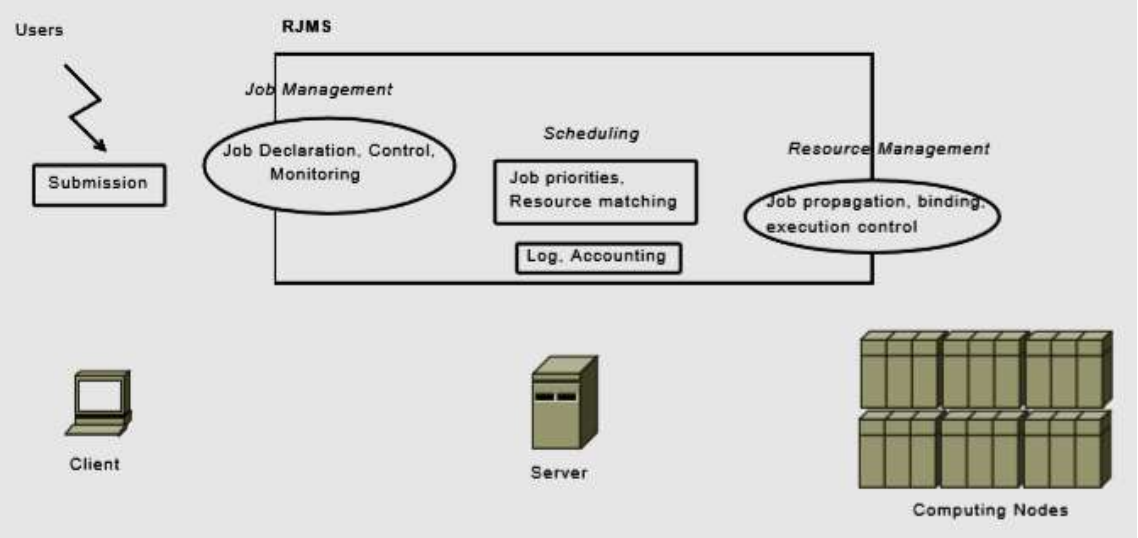
It is a preliminary step to scheduling

- Resources filtering
- Resources sorting
- Allows the user to have special needs: memory, architecture, special hosts, OS, workload

The below table describes the various RJMS subsystems

RJMS subsystems	Principal Concepts	Advanced Features
Resource Management	-Resource Treatment (hierarchy, partitions,..) -Job Launcing, Propagation, Execution control -Task Placement (topology,binding,..)	- High Availability - Energy Efficiency - Topology aware placement
Job Management	-Job declaration (types, characteristics,...) -Job Control (signaling, reprioritizing,...) -Monitoring (reporting, visualization,..)	- Authentication (limitations, security,..) - QOS (checkpoint, suspend, accounting,...) - Interfacing (MPI libraries, debuggers, APIs,..)
Scheduling	-Scheduling Algorithms (builtin, external,..) -Queues Management (priorities,multiple,..)	- Advanced Reservation - Application Licenses

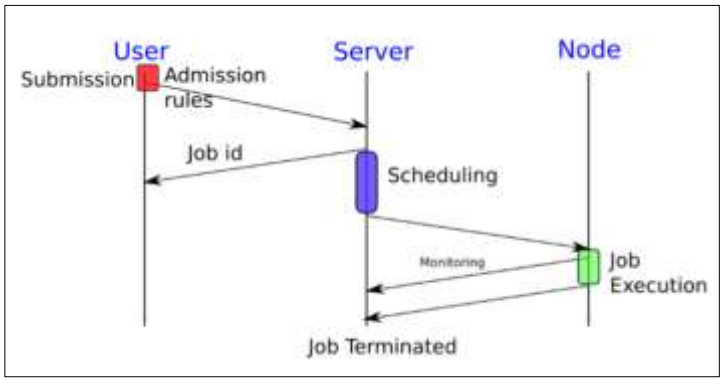
RMJS Central Organization



Scheduling policies in cluster computing

Scheduling Policy	Description
FIFO	jobs are treated with the order they arrive.
Backfill	fill up empty wholes in the scheduling tables without modifying the order or the execution of previous submitted jobs.
Gang Scheduling	multiple jobs may be allocated to the same resources and are alternately suspended/resumed letting only one of them at a time have dedicated use of those resources, for a predefined duration.
TimeSharing	multiple jobs may be allocated to the same resources allowing the sharing of computational resources. The sharing is managed by the scheduler of the operating system
Fairshare	take into account past executed jobs of each user and give priorities to users that have been less using the cluster.
Preemption	suspending one or more "low-priority" jobs to let a "high-priority" job run uninterrupted until it completes

Job life cycle

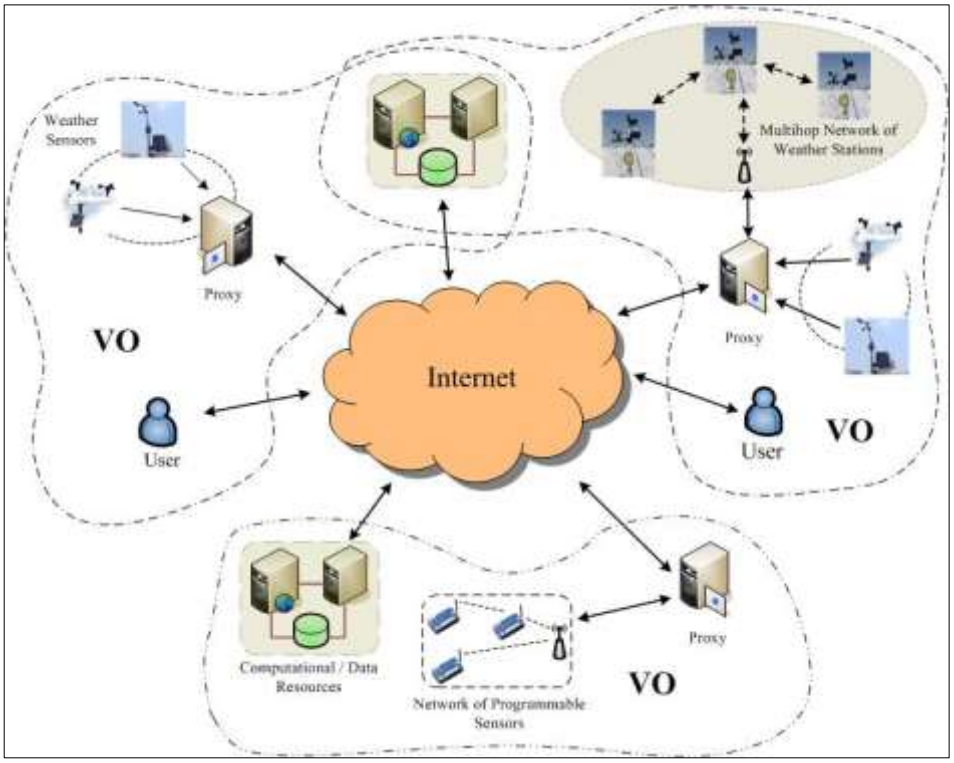


Applications of CLUSTER COMPUTING

1. Educational resources
2. Commercial sectors for industrial promotion
3. Medical research

GRID COMPUTING: Description and applications

Grid computing is the segregation of resources from multiple sites so as to solve a problem that can't be solved by using the processing of a single computer. It employs use of **multiple clusters that are loosely coupled**, **heterogeneous and are geographically dispersed**. Here individual user gets access to the resources (like processors, storage, data etc.) on demand with little or no knowledge of the fact that where those resources are physically located. It is more popularly known as a collection of servers that are bound together to attack a single problem.



Main goals of grid computing:

- Grid computing is concerned about sharing, collecting, hosting and providing services to various consumers
- Exploiting underutilized computational resources (reducing costs)
- Allowing very large computation through virtual parallel machines (speed up)
- The essential mode to implement the Grid is through Virtual Organizations (VO)

Resource management and scheduling

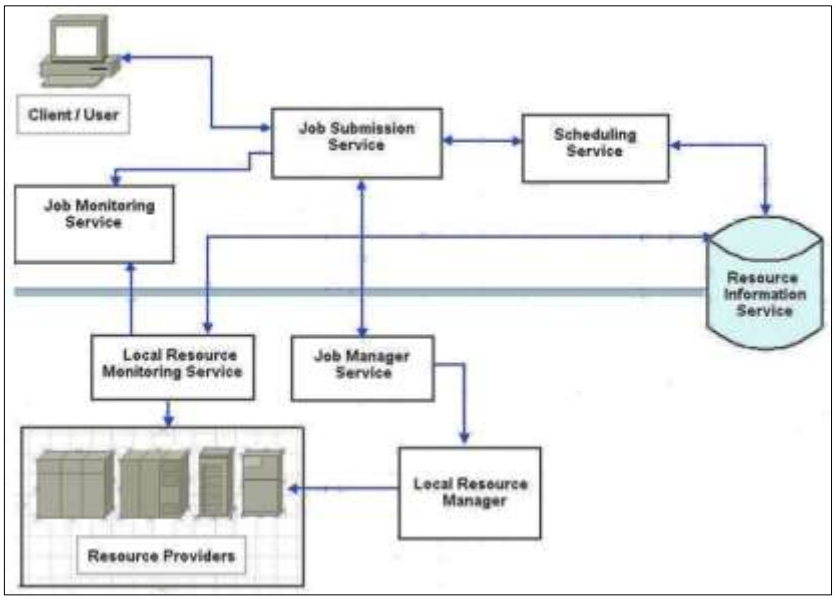
The model in grid computing can be peer to peer or client/server. Moreover, the same resource may be used in different ways.

The main challenges to be handled in grid resource management are:

- Satisfactory end-to-end performance through multiple domains
- Availability of computational resources
- Handle of conflicts between common resources demand
- Fault tolerance
- Inter domain compatibility (P2P)

In the grid computing network, we have: Computing power, disk space, memory space, Network bandwidth and software which must be managed according to the demand from the user.

Stages in resource management



- **Phase 1: Resource Discovery**
  - Find available resources
- **Phase 2: Systems Selection**
  - Allocate the resources
- **Phase 3: Job Execution**
  - Run the job
  - Log the resource usage
  - Release the resources
  - Charge the user

Local resource management system (Resource Level)

- Basic resource management unit
- Provides low level resource allocation and scheduling

Global resource management system (Grid Level)

- Coordinates the local resource management systems within multiple or distributed sites
- Provides high-level functionalities to efficiently use resources:
  - Job submission
  - Resource discovery and selection
  - Authentication, authorization, accounting
  - Scheduling
  - Job monitoring
  - Fault tolerance

Scheduling scenario

- A finite set of n jobs
- Each job consists of a chain of operations
- A finite set of m machines
- Each machine can handle at most one operation at a time
- Each operation needs to be processed during an uninterrupted period of a given length on a given machine
- Purpose is to find a schedule, that is, an allocation of the operations to time intervals to machines, that has minimal length.

Algorithms for scheduling computational resources belonging to the Grid Computing Infrastructure **are usually based on Heuristic methodologies**

Problems in Grid Scheduling

1. **Grid schedulers do not own resources themselves**
   a. they have to negotiate with autonomous local schedulers
   b. authentication/multi-organizational issues
2. **Grid schedulers have to interface to different local schedulers**
   a. some may have support for reservations, others are queuing-based
   b. some may support check pointing, migration, etc.

3. **Structure of applications**
    a. many different structures (parallel, PSAs, workflows, database, etc.)
    b. need for application adaptation modules
4. **Failures**
    a. Must monitor the progress of applications/sanity of systems
    b. only thing we know to do upon failures: restart (possibly from a checkpoint)
5. **Performance metric**
    a. cannot calculate the turn-around time

Applications of GRID COMPUTING

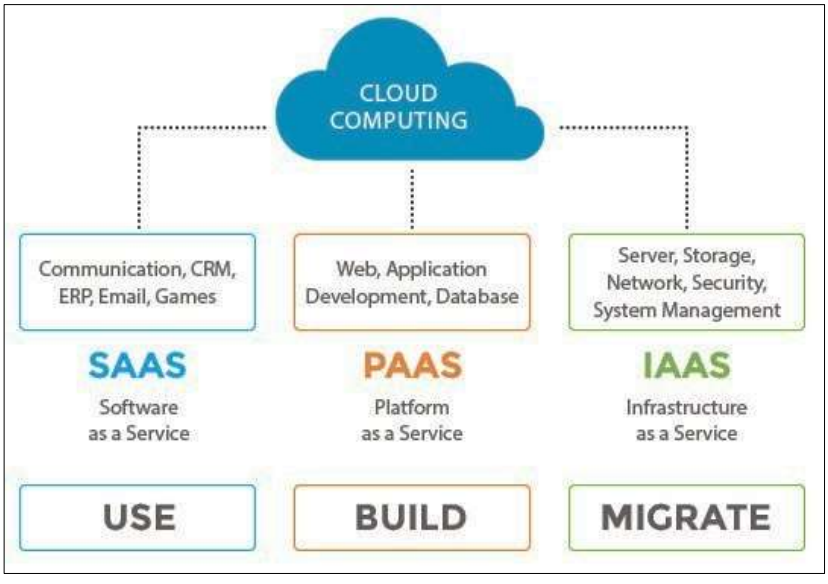
1. Predictive Modeling and Simulations
2. Engineering Design and Automation
3. Energy Resources Exploration
4. Medical, Military and Basic Research
5. Visualization

CLOUD COMPUTING: Description and applications

Cloud computing is the computing paradigm which provides **large pool of dynamical scalable and virtual resources as a service on demand.** The main principle behind cloud computing model is to offer computing, storage, and software as a service or as a utility. We just need internet to use these utilities.

"Cloud is a parallel and distributed computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements (SLA) established through negotiation between the service provider and consumers."

Software as a Service(SaaS): To complete the business tasks Business user uses the SaaS Model to delivered the web application. SaaS provides Automation, Customer Relationship Model, Website testing and many more services.



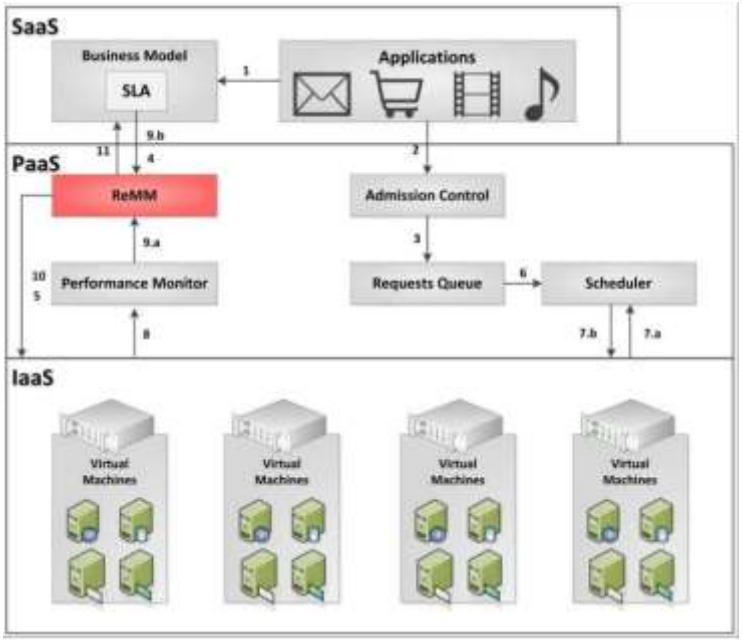
Platform as a Service(PaaS): Developers and Deplorers uses the PaaS model to create or deploy application and services for users. Application test, development, integration and deployment, kind of services provided by the PaaS model.

Infrastructure as a Service(IaaS): System manager uses the IaaS model to create platform for service and application testing. Virtual machine, operating system message queue, network, storage, CPU, memory, backup services are available in IaaS Model.

ReMM—Resource Management Module (a privately implemented module by researcher)

The ReMM concerns the way the available resources are handled during the execution time with regard to the QoS metrics defined in the SLA. It is an outline for a resource management module for cloud applications in which both

horizontal and vertical scalability can be applied dynamically, thus leading to a change in price. The most common type of scaling is horizontal scaling which involves the allocation and release of virtual machines. The other type is vertical scaling which either increases or reduces the computing resources (vCPU, memory, disk, etc.) of one or more instances.



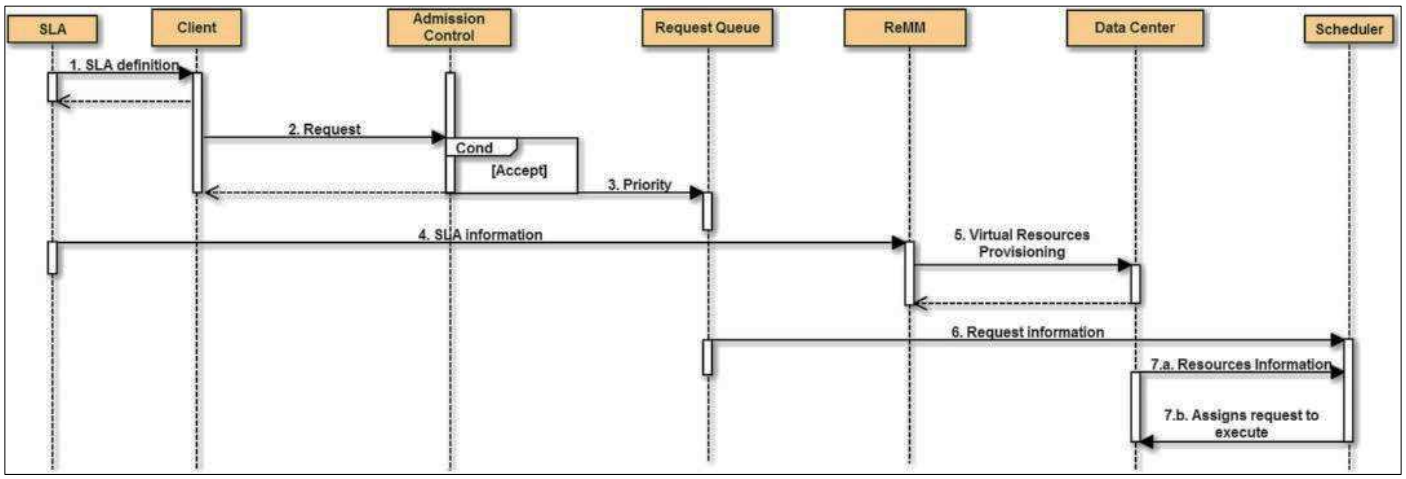
1. A **client's request** will be answered by a provider, which uses three layers and can provide different services: **application layer** (Software as a Service—SaaS), **platform layer** (Platform as a Service—PaaS) and **infrastructure layer** (Infrastructure as a Service—IaaS).
2. The **application layer** handles all the application services that are offered to the clients.
3. The **platform layer** includes mapping and scheduling policies which are designed to translate the clients' QoS requirements to infrastructure level parameters and allocating virtual machines to meet their requests.
4. The **infrastructure layer** carries out the **initiation and removal of VMs** with specific resource configurations for the client in a transparent way.

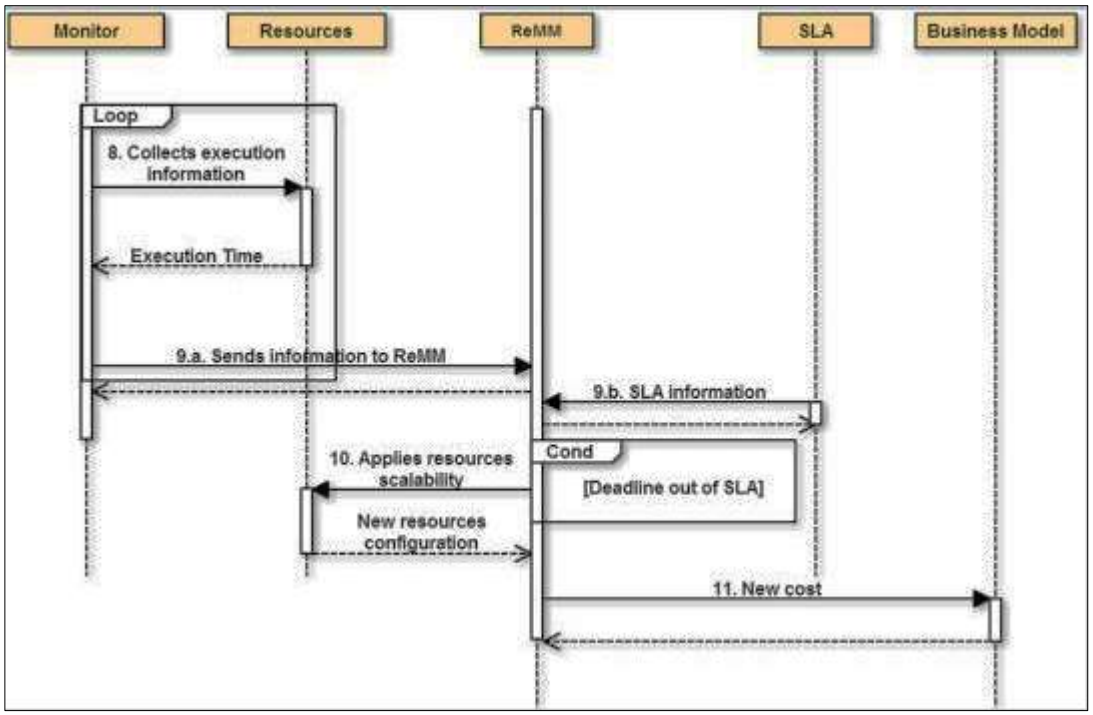
The client and provider must negotiate a SLA, by defining the QoS metrics and the contract details. After this procedure, different clients request different types of services from a provider. During the system overload, for example, the service provider can decide either to reject the new requests or violate the SLA. The SLA violation should result in penalties for the provider.

5. The **ReMM will provide the virtual resources based on the information defined in the SLA**, and place them in the physical resources.
6. **After the resources allocation**, the Scheduler forwards the requests so that they can be run on resources using scheduling policies.
7. At intervals of time, the **Performance Monitor** collects information about the system performance and about the request execution and sends them to ReMM. This information is compared with the QoS information available in the contract and, **if the results are not in accordance with the SLA**, the ReMM dynamically adjusts the amount of resources (horizontally or vertically) in an attempt to ensure the SLA.

Other modules can be included in the environment described, such as a module for prediction, analysis and load balancing, or new scheduling policies that are designed, for example, to assist in energy saving.

The **sequence diagrams in Figs 2 and 3** show the interactions carried out in response to the client request and the analyses performed to determine whether the resources should be changed or not, and the proportional this has on costs.





Applications of CLOUD COMPUTING

1. Banking
2. Insurance
3. Weather Forecasting
4. Space Exploration
5. SaaS, PaaS, IaaS

COMPARITIVE STUDY: Cluster, grid and cloud computing

DHRUV GARG 16 BCE 1190

COMPARITIVE STUDY: Cluster, grid and cloud computing

Cluster Computing	Grid Computing	Cloud Computing
Characteristics of Cluster computing 1. Tightly coupled systems 2. Single system image 3. Centralized Job management & scheduling system	Characteristics of Grid computing 1. Loosely coupled (Decentralization) 2. Diversity and Dynamism 3. Distributed Job Management & scheduling	Characteristic of cloud computing 1. Dynamic computing infrastructure 2. IT service-centric approach 3. Self-service based usage model 4. Minimally or self-managed platform 5. Consumption-based billing
In cluster computing, a bunch of similar (or identical) computers are hooked up locally (in the same physical location, directly connected with very high speed connections) to operate as a single computer	In grid computing, the computers do not have to be in the same physical location and can be operated independently. As far as other computers are concerned each computer on the grid is a distinct computer.	In cloud computing, the computers need not to be in the same physical location.
The cluster computers all have the same hardware and OS.	The computers that are part of a grid can run different operating systems and have different hardware	The memory, storage device and network communication are managed by the operating system of the basic physical cloud units.
The whole system (all nodes) behaves like a single system view and resources are managed by centralized resource manager.	Every node is autonomous i.e. it has its own resource manager and behaves like an independent entity	Every node acts as an independent entity
More than 2 computers are connected to solve a problem	A large project is divided among multiple computers to make use of their resources.	It does just the opposite. It allows multiple smaller applications to run at the same time.
Commodity computers	Size or scalability is 1000s	Size or scalability is 100s to 1000s
Single Ownership	Multiple Ownership	Single Ownership
Dedicated, high-end with low latency and high bandwidth Interconnection Network	Mostly Internet with high latency and low Bandwidth Interconnection Network	Dedicated, high-end with low latency and high Bandwidth Interconnection Network
User management is centralized	User management is decentralized and also virtual organization (VO)-based	User management is centralized or can be delegated to third party
Resource management is centralized	Resource management is distributed	Resource management is centralized/distributed
Stable and guarantee capacity	Varies, but high capacity	Provisioned on demand capacity
Failure management (Selfhealing) is limited (often failed tasks/applications are restarted).	Failure management (Selfhealing) is limited (often failed tasks/applications are restarted).	Strong support for failover and content replication. VMs can be easily migrated from one node to other.

RESULT:

Thus the characteristics of the three paradigms have been explained in detail, and the comparison has been made.