CS8791-CLOUD COMPUTING

UNIT-I INTRODUCTION

CHAPTER 1 INTRODUCTION

CLOUD COMPUTING

Introduction to Cloud Computing

Computing as a service has seen a phenomenal growth in recent years. The primary motivation for this growth has been the promise of reduced capital and operating expenses, and the ease of dynamically scaling and deploying new services without maintaining a dedicated compute infrastructure. Hence, cloud computing has begun to rapidly transform the way organizations view their IT resources. From a scenario of a single system consisting of single operating system and single application, organizations have been moving into cloud computing, where resources are available in abundance and the user has a wide range to choose from. Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with service provider interaction or minimal management effort. Here, the end-users need not to know the details of a specific technology while hosting their application, as the service is completely managed by the Cloud Service Provider (CSP). Users can consume services at a rate that is set by their particular needs. This on-demand service can be provided any time. CSP would take care of all the necessary complex operations on behalf of the user. It would provide the complete system which allocates the required resources for execution of user applications and management of the entire system 2 flow.

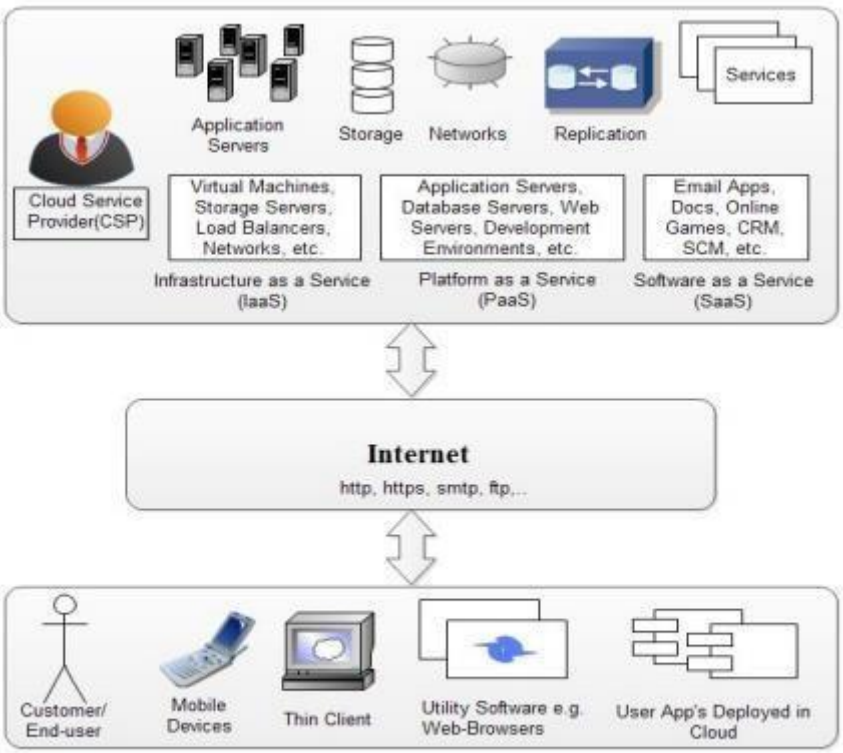


Fig. 1.1 Cloud Computing Architecture.

Cloud computing consists of three distinct types of computing services delivered remotely to clients via the internet. Clients typically pay a monthly or annual service fee to providers, to gain access to systems that deliver software as a service, platforms as a service and infrastructure as a service to subscribers. Clients who subscribe to cloud computing services can reap a variety of benefits, depending on their particular business needs at a given point in time. The days of large capital investments in software and IT infrastructure are now a thing of the past for any enterprise that chooses to adopt the cloud computing model for procurement of IT services. The ability to access powerful IT resources on an incremental basis is leveling the playing field for small and medium sized organizations, providing them with the necessary tools and technology to compete in the global marketplace, without the previously requisite investment in on premise IT resources. Clients who subscribe to computing services delivered via the "cloud"• are able to greatly reduce the IT service expenditures for their organizations; and gain access to more agile and flexible enterprise level computing services, in the process.

Types of service in cloud computing

SAAS(Software as a Service)

Saas (Software as a Service) provides clients with the ability to use software applications on a remote basis via an internet web browser. Software as a service is also referred to as "software on demand". Clients can access SaaS applications from anywhere via the web because service providers host applications and their associated data at their location. The primary benefit of SaaS, is a lower cost of use, since subscriber fees require a much smaller investment than what is typically encountered under the traditional model of software delivery. Licensing fees, installation costs, maintenance fees and support fees that are routinely associated with the traditional model of software delivery can be virtually eliminated by subscribing to the SaaS model of software delivery. Examples of SaaS include: Google Applications and internet based email applications like Yahoo! Mail, Hotmail and Gmail.

PAAS(Platform as a Service)

PaaS (Platform as a Service) provides clients with the ability to develop and publish customized applications in a hosted environment via the web. It represents a new model for software development that is rapidly increasing in its popularity. An example of PaaS is Salesforce.com. PaaS provides a framework for agile software development, testing, deployment and maintenance in an integrated environment. Like SaaS, the primary benefit of PaaS, is a lower cost of use, since subscriber fees require a much smaller investment than what is typically encountered when implementing traditional tools for software development, testing and deployment. PaaS providers handle platform maintenance and system upgrades, resulting in a more efficient and cost effective solution for enterprise software development.

IAAS(Infrastructure as a Service)

IaaS (Infrastructure as a Service) allows clients to remotely use IT hardware and resources on a "pay-as-you-go"• basis. It is also referred to as HaaS (hardware as a service). Major IaaS players include compa nies like IBM, Google and Amazon.com. IaaS employs virtualization, a method of creating and managing infrastructure resources in the "cloud"• . IaaS provides small start up firms with a major advantage, since it allows them to gradually expand their IT infrastructure without the need for large capital investments in hardware and peripheral systems.

DEFINITION OF CLOUD

Cloud Computing is the use of hardware and software to deliver a service over a network (typically the Internet). With cloud computing, users can access files and use applications from any device that can access the Internet. An example of a Cloud Computing provider is Google's Gmail. Cloud Computing lets you store and access your applications or data over remote computers instead of your own computer.

Cloud Computing can be defined as delivering computing power (CPU, RAM, Network Speeds, Storage OS software) a service over a network (usually on the internet) rather than physically having the computing resources at the customer location.

Example: AWS, Azure, Google Cloud

Why Cloud Computing?

With increase in computer and Mobile user's, data storage has become a priority in all fields. Large and small scale businesses today thrive on their data & they spent a huge amount of money to maintain this data. It requires a strong IT support and a storage hub. Not all businesses can afford high cost of in-house IT infrastructure and back up support services. For them Cloud Computing is a cheaper solution. Perhaps its efficiency in storing data, computation and less maintenance cost has succeeded to attract even bigger businesses as well.

Cloud computing decreases the hardware and software demand from the user's side. The only thing that user must be able to run is the cloud computing systems interface software, which can be as simple as Web browser, and the Cloud network takes care of the rest. We all have experienced cloud computing at some instant of time, some of the popular cloud services we have used or we are still using are mail services like Gmail, Hotmail or Yahoo etc.

While accessing e-mail service our data is stored on cloud server and not on our computer. The technology and infrastructure behind the cloud is invisible. It is less important whether cloud services are based on HTTP, XML,

Ruby, PHP or other specific technologies as far as it is user friendly and functional. An individual user can connect to cloud system from his/her own devices like desktop, laptop or mobile. Cloud computing harnesses small business effectively having limited resources, it gives small businesses access to the technologies that previously were out of their reach.

Benefits of Cloud Computing

First of all, 'cloud' is just a metaphor to describe the technology. Basically, cloud is nothing but a data center filled with hundreds of components like servers, routers, and storage units. Cloud data centers could be anywhere in the world; also you can access it from anywhere with an Internet-connected device. Why do people use it? Because of the following benefits it has:

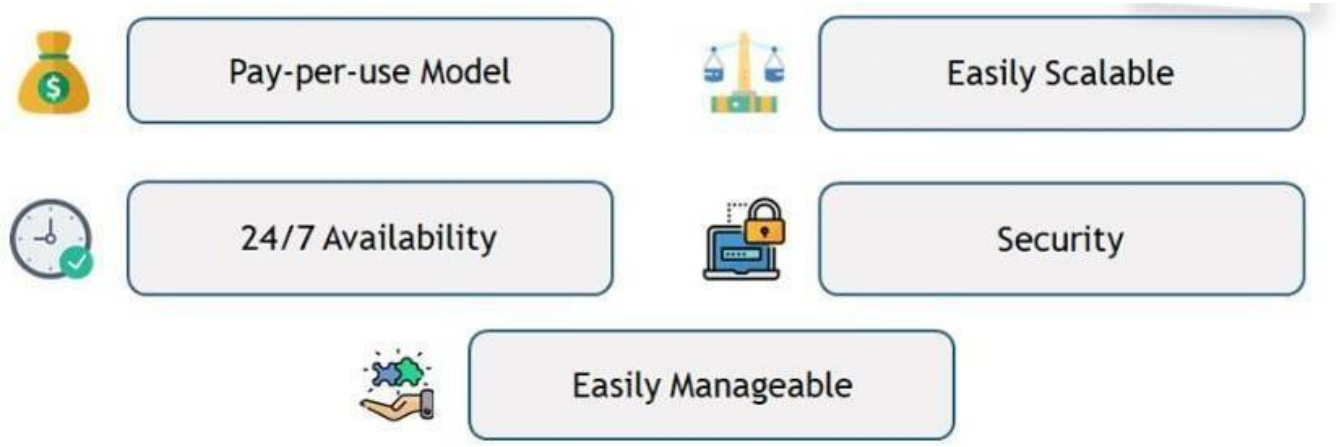


Fig 1.2 Benefits of cloud computing

Pay-per-use Model: You only have to pay for the services you use, and nothing more!

24/7 Availability: It is always online! There is no such time that you cannot use your cloud service; you can use it whenever you want.

Easily Scalable: It is very easy to scale up and down or turn it off as per customers' needs. For instance, if your website's traffic increases only on Friday nights, you can opt for scaling up your servers that particular day of the week and then scaling down for the rest of the week.

Security: Cloud computing offers amazing data security. Especially if the data is mission-critical, then that data can be wiped off from local drives and kept on the cloud only for your access to stop it ending up in wrong hands.

Easily Manageable: You only have to pay subscription fees; all maintenance, up-gradation and delivery of services are completely maintained by the Cloud Provider. This is backed by the Service-level Agreement(SLA).

EVOLUTION OF CLOUD COMPUTING

A recent sensation in the realm of outsourcing is called Cloud Computing. Cloud is a huge collection of effortlessly approachable imaginary like utilities that can be used and accessed from anywhere, (for example s/w, h/w, advancement operating environments and applications). These operating environments and applications could be alterably re-designed to acclimate to a varying burden, permitting likewise for best environment utilization. These environments and facilities are ordinarily known to be a per utilization payment arrangement where in insurances are guaranteed by the service issuer by method of altered service level agreements.

While the happening to contemporary workstation systems administration happened inside mid1970s, not any talk of something remotely looking like a thought like "cloud computing" occurred until at long last in the ballpark of an a few years after the fact in 8f when John Back of Sun Microsystems begat the noteworthy motto, "The real system is The true machine." As prophetic similarly Sun was a while back, fittings (essential Figure and systems administration) had been none, of these influential none, of these commoditized enough to attain this vision around then. Cloud computing was still leastways a decade off. Meanwhile, Sun's Unix-delightful working framework and servers turned into the "new iron," swapping mainframes that been around for numerous eras.

Sun's machines utilized open systems administration models, for example TCP/IP. This empowered projects running on only one machine to identify with running systems on different models. Such requisitions ordinarily emulated the customer server building design model. Around this time period, Sir Tim Berners-Lee prescribed the thought connected with imparting learning spared on numerous hosting space to be made accessible to the planet through customer machines. Documents might be placed holding hypertext, course book with metadata made up of area data as a referrals for the thing portrayed by means of that content.

That conceived an offspring towards the Common Gateway Interface. Common gateway interface presented outer provisions in the web server to system client enter notwithstanding process or get the Html records. The "web application" had come. We've now minute voyage from the genuine mid-1980s on the early to mid-1990s. Common gateway interface made alert content achievable, which in flip realized customization. Common gateway interface's cutoff points were being likewise arrived at quickly when then transformed into comparative however more versatile results. Winning static archives paled in contrasting with being able to create Html alertly on-investment on clients. It was conceivable to make web keeping money: a singular and everybody furthermore could log into and view their novel ledgers; purchasers could potentially buy items internet, taking mail request of size to a completely new level. A standout amongst the most powerful of online stores has been and it is still to this time, Amazon. It's most likely a brilliant amaze that Rain woods came into staying not long after the production of the true common gateway interface standard. From here in the center 1990s, they planned one of the best and most massive system of workstation frameworks by the first 2000s. They purchased enough computing assets to allow their business to withstand the true organize activity require for the occasion acquiring season. Concurring on their whitepaper: "By 2006, we had utilized in excess of a couple of years and hundreds with respect to a large number of cash building and administering the vast scale, strong, and proficient It foundation that fueled your operation of one of numerous planet's biggest online retail stages." Your quandary: what does this mechanical assembly do throughout the sunny season? It sat unmoving. Why not "sublease" this apparatus to attempt and recover expenses? All around mid2006, these individuals started Amazon Web Services, promoting their computing power to anybody. Other substantial legitimately known innovation organizations with computing assets joined the in general amusement that truly carries us to today.

Despite the fact that the exact history of cloud computing is not the existing (the first business and customer cloud computing administrations sites – salesforce.com in addition to Google, were presented in 1999), the story is trussed straight to the improvement of the Internet and business engineering, since cloud computing is the method to the issue connected with how the Internet can help fortify business innovation. Business innovation

has a drawn out and intriguing history, one that is almost as long on the grounds that business itself, yet the specific advancements that about straight impacted the of cloud computing begin with The true rise of workstations as suppliers of genuine business results.

In spite of the fact that machines had been pretty nearly for some period, the state of the innovation and the convention of the fifties made the best environment for start to be processed rapidly. Machines have been fundamentally huge adding machines that utilized hit cards to work with figuring's, so however they were suitable, progressions could plainly come in. The social setting was molded by method of Cold War interest with the utopia that could be made utilizing engineering, besides since newly discovered thriving to store it. These two elements implied individuals wound up both eager and likewise fit to purchase headways. The exact first useful microchip got it's begin in the past due 1950s, and once workstations could accomplish more intricate computations, individuals began building systems for business requisitions. The exact first normal office machine work is done in 1951 by the true Leo workstation, which was made to manage the real overnight handling determinations, payroll, stock and additionally other managerial assignments for J. Lyons & Co., a providing food in addition to sustenance assembling association. This could be recognized the initially incorporated administration data framework. Just about as fast as machines had been showing their pragmatic use good to go, organizations began examining how to administration littler organizations. Inside 1959, IBM discharged the 1401 model, which gave minor undertakings access to actualities transforming machines. Your 1401 was popular to the point that by the mid-60s, about 50% of the planet's machine frameworks were 1401-character frameworks.

The internet's "youth" Whereby it came to be clear that Arpanet was a decently major ordeal and some huge workstation organizations were made In the nineteen seventies, the thoughts and likewise components that had been prescribed in the 50s and 60s were being created decisively. Additionally, a large portion of the world's greatest machine organizations were begun, and the internet was conceived. Inside 1971, Intel, set up in the previous decade, acquainted the earth with the precise first microchip, and Apple organization architect Ray Tomlinson composed a system that permitted individuals to send correspondences starting with one machine then onto the next, from this point forward sending the essential message that a great deal of individuals might recognize similarly message.

Then, Bill Gates and Robert Allen established Microsoft in 1974; however Steve Wozniak and additionally Steve Jobs started Apple Computers inside 1976 and divulged the Apple Two in the indistinguishable year. All your while, the All of us Department of Protection had been making Arpanet into Internet, and in 1976, Xerox's Robert Metcalfe indicated the idea of Ethernet. Later inside the decade, CompuServe Information Services and The definitive source both went on the internet in 1979, pre-figuration the focus if the internet might be supervised by and used through business administration suppliers. The 80s introduced the introductory major, worldwide mouthpiece blast in Pcs. Through 1980, there had been more than 5 different million Pcs being used worldwide, however normally these were expected for business or government utilization. So in 1981, IBM position the first Pc on the industry, and in 1982, Microsoft started certificate MS DOS, this working framework that, as a result of gigantic scale marketing deliberations by Microsoft organization, most Pc frameworks might run along. At that point, in place of your worldwide dystopia, 1984 carried the first Macintosh Pc, the establishing joined with Dell workstation by Michael Dell and William Gibson's begetting of the statement 'theinternet'.

The plant seeds were being seeded for the increment of the internet. The internet's global starting Where the Internet as a spot for both business and correspondence carried its own weight. The specific 1990s joined the planet in an unmatched way, beginning together with Cern's ease the World Wide-cut Web for general (that is, non-business) use in 1991. In Michael became bonkers, a browser reputed to be Mosaic permitted pictures to be demonstrated on Internet, and private organizations were permitted to work with Internet despite anything that might have happened before, as well. When firms were on the web, they started to assume the business potential outcomes that accompanied having the ability to achieve the planet in an immediate, and a percentage of the greatest players online were established.

Marc Andreessen and John Clark established Netscape in 94' and none excessively early, since 1995 sawing machine internet activity gave up to business endeavors like Netscape. In the meantime, stalwarts of the true internet Amazon.com and likewise eBay were started by Jeff Bezos and Pierre Omidyan, severally. The internet's "adulthood" notwithstanding cloud computing ascent Where the dab com house of cards blasts as a pimple notwithstanding cloud computing goes to the real fore The remainder of the nineties and starting with the 2000s were an incredible opportunity to find or put resources into an internet-based organization. Cloud computing had the right environment to lose, as multi-leaseholder architectures, profoundly overarching fast data transfer capacity and regular programming interoperability details were created in this specific time. Salesforce.net appeared in late 90s and was the essential webpage to convey business requisitions from the "typical" site – what on earth is presently called cloud computing.

The unadulterated confidence of this period prompted the specific spot com blast, where internet based organizations, backed by evidently very nearly unending wander capital and likewise excessively sure expectations, developed quickly. The majority of these sites trusted they can run at a misfortune briefly, then accuse of respect to administrations later, which implied they have been running on venture capital and acquiring no wage at wholly. Amazon and Search motors both neglected to work at a benefit in their first years, however it was in light of the fact that they wound up using cash on marketing deliberations or upgrading their innovation. The dab com bubble hit the top on tenth March 2000, then blast over the accompanying weekend as major high-tech stockholders like Dell and Coregonus artedi sold off an incredible arrangement of their stock. To prepare could have carried to the failure of the house of cards, incorporating the opposition to -syndication controlling towards Microsoft (which, however, not uncovered until they begin of April, had as of recently been generally foreseen).

Different explanations incorporate poor on-line (a) takings from the 1999 Christmas time of year and Y2K all things considered not in the planet-outcome way numerous individuals anticipated. Rather, companies used a bunches of cash redesigning frameworks and supplies, so when the day passed on without occasion, organizations quit using cash to blanket the inconvenience of upgrading an incredible arrangement. This implied that they quit exchanging, false procuring stops and generally discovered approaches to curtail financial spending. Regardless, research shows that half of us speck comes made due until 2004, so rather than being headed out from business by the unfolding of the belch, organizations either do well or were not reasonable. Still, to stay to survive, enterprises needed to rethink or refine their specific plans of action and what you offered to purchasers. Numerous more current companies chose to indulgence administrations that watched the internet as a basic part of the true administration, as opposed to like a medium to area requests or correspond with clients.

Amazon.com presented Amazon Web Services in 2004. This gave clients the capacity with a specific end goal to store information and hang a huge amount of people to work with exceptionally little obligations, (for

example Hardware Turk), around some different administrations. Face book had been established in '04, revolutionizing the methodology clients impart and the way they store their own particular information (their photos and film), inadvertently making the cloud an individual administration.

In 2006, Amazon online unfolded its cloud administrations. To begin with had been its Elastic Compute cloud (Ec2), which allowed individuals to memory access machines and running their own particular programming on them, altogether on the cloud. At that point they presented Simple Storage Service (S3). This introduced the pay-seeing that-you-head off model to both purchasers and the business sector overall, and it has fundamentally come to be standard practice at present. Salesforce.com then started force.com in 2007. That stage as an administration let associations' engineers construct, retailer and run the sum of the applications and sites they required running their business in the cloud. Google Apps started in 2009, permitting individuals to make and store paperwork truly in the genuine cloud.

Most as of late, cloud computing organizations have been pondering how they might make their stock all the more join. In 2010 Salesforce.com started the cloud-based database at Database.com planned for engineers, denoting the advancement of could computing administrations that might be utilized on any unit, run on practically any stage and composed in diverse modifying dialect. Obviously, the long run of the internet and cloud computing have in prior times demonstrated hard to compute, however so farseeing as organizations strive to unite the globe and serve in which joined planet with new ways, there'll dependably be a need for both the internet and cloud computing.

UNDERLYING PRINCIPLES OF PARALLEL AND DISTRIBUTED COMPUTING

Parallel Computing

Cloud computing is intimately tied to parallel and distributed processing. Cloud applications are based on the client–server paradigm. Such applications run multiple instances of the service and require reliable and in-order delivery of messages. Parallel computing is a term usually used in the area of High Performance Computing (HPC).It specifically refers to performing calculations or simulations using multiple processors. Supercomputers are designed to perform parallel computation. These system do not necessarily have shared memory (as incorrectly claimed by other answers

What is Parallel Computing?

Parallel computing is a type of computation in which many calculations or execution of processes is carried out simultaneously. Whereas, a distributed system is a system whose components are located on different networked computers which communicate and coordinate their actions by passing messages to one another. Thus, this is the fundamental difference between parallel and distributed computing.

Parallel computing is also called parallel processing. There are multiple processors in parallel computing. Each of them performs the computations assigned to them. In other words, in parallel computing, multiple calculations are performed simultaneously. The systems that support parallel computing can have a shared memory or distributed memory. In shared memory systems, all the processors share the memory. In distributed memory systems, memory is divided among the processors.

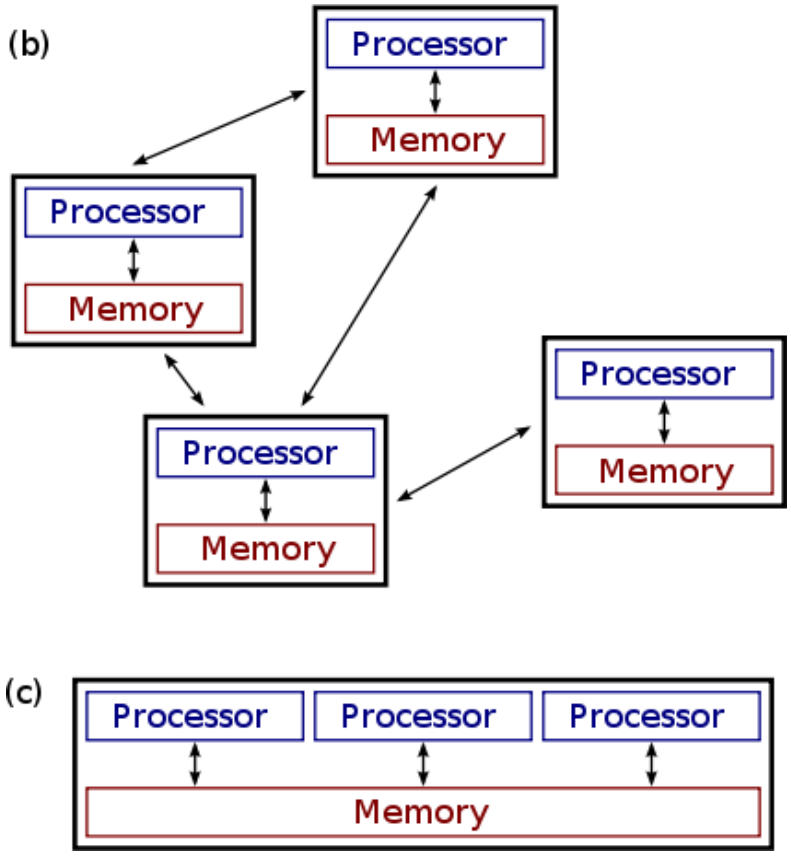
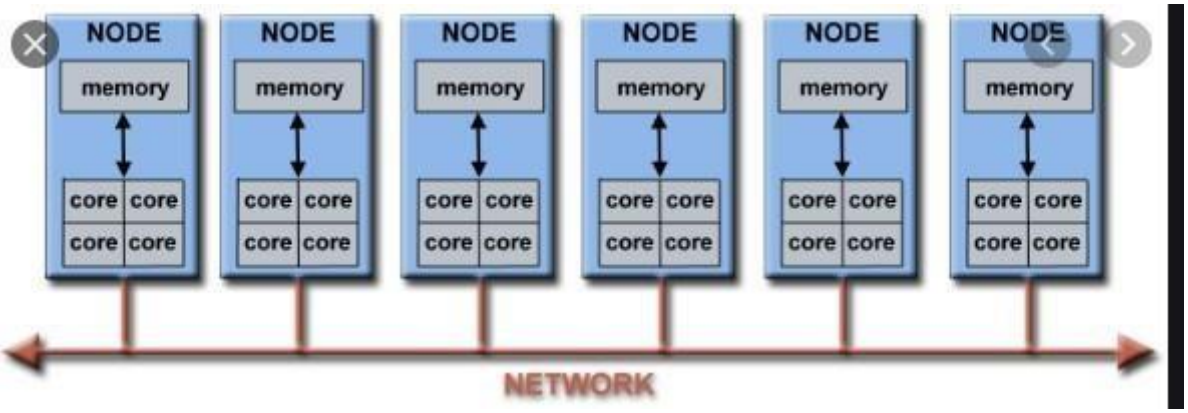


Figure 1.5 Parallel computing architecture

There are multiple advantages to parallel computing. As there are multiple processors working simultaneously, it increases the CPU utilization and improves the performance. Moreover, failure in one processor does not affect the functionality of other processors. Therefore, parallel computing provides reliability. On the other hand, increasing processors is costly. Furthermore, if one processor requires instructions of another, the processor might cause latency.



In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem:

- A problem is broken into discrete parts that can be solvedconcurrently
- Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different processors
- An overall control/coordination mechanism is employed

Parallel Computing

It is the use of multiple processing elements simultaneously for solving any problem. Problems are broken down into instructions and are solved concurrently as each resource which has been applied to work is working at the same **time.**

Advantages of Parallel Computing over Serial Computing are as follows:

- It saves time and money as many resources working together will reduce the time and cut potential costs.
- It can be impractical to solve larger problems on Serial Computing.
- It can take advantage of non-local resources when the local resources are finite.

Serial Computing 'wastes' the potential computing power, thus Parallel Computing makes better work of hardware.

Types of Parallelism:

Bit-level parallelism: It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data.

Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.

Instruction-level parallelism: A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.

Task Parallelism: Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform execution of sub tasks concurrently.

Why parallel computing?

- The whole real world runs in dynamic nature i.e. many things happen at a certain time but at different places concurrently. This data is extensively huge to manage.
- Real world data needs more dynamic simulation and modeling, and for achieving the same, parallel computing is the key.
- Parallel computing provides concurrency and saves time and money.
- Complex, large datasets, and their management can be organized only and only using parallel computing's approach.
- Ensures the effective utilization of the resources. The hardware is guaranteed to be used effectively whereas in serial computation only some part of hardware was used and the rest rendered idle.

Also, it is impractical to implement real-time systems using serial computing.

Applications of Parallel Computing:

- Data bases and Data mining.
- Real time simulation of systems.
- Science and Engineering.
- Advanced graphics, augmented reality and virtual reality.

Limitations of Parallel Computing:

- It addresses such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve.

- The algorithms must be managed in such a way that they can be handled in the parallel mechanism.
- The algorithms or program must have low coupling and high cohesion.
- But it's difficult to create such **programs.**
- More technically skilled and expert programmers can code a parallelism based program well.

Future of Parallel Computing: The computational graph has undergone a great transition from serial computing to parallel computing. Tech giant such as Intel has already taken a step towards parallel computing by employing multicore processors. Parallel computation will revolutionize the way computers work in the future, for the better good. With all the world connecting to each other even more than before, Parallel Computing does a better role in helping us stay that way. With faster networks, distributed systems, and multi-processor computers, it becomes even more necessary.

What is Distributed Computing?

Distributed computing divides a single task between multiple computers. Each computer can communicate with others via the network. All computers work together to achieve a common goal. Thus, they all work as a single entity. A computer in the distributed system is a node while a collection of nodes is a cluster.

A Distributed System is composed of a collection of independent physically (and geographically) separated computers that do not share physical memory or a clock. Each processor has its own local memory and the processors communicate using local and wide area networks. The nodes of a distributed system may be of heterogeneous architectures.

A Distributed Operating System attempts to make this architecture seamless and transparent to the user to facilitate the sharing of heterogeneous resources in an efficient, flexible and robust manner. Its aim is to shield the user from the complexities of the architecture and make it appear to behave like a timeshared centralized environment.

Distributed Operating Systems offer a number of potential benefits over centralized systems. The availability of multiple processing nodes means that load can be shared or balanced across all processing elements with the objective of increasing throughput and resource efficiency. Data and processes can be replicated at a number of different sites to compensate for failure of some nodes or to eliminate bottlenecks. A well designed system will be able to accommodate growth in scale by providing mechanisms for distributing control and data.

Communication is the central issue for distributed systems as all process interaction depends on it. Exchanging messages between different components of the system incurs delays due to data propagation, execution of communication protocols and scheduling. Communication delays can lead to inconsistencies arising between different parts of the system at a given instant in time making it difficult to gather global information for decision making and making it difficult to distinguish between what may be a delay and what may be a failure.

Fault tolerance is an important issue for distributed systems. Faults are more likely to occur in distributed systems than centralized ones because of the presence of communication links and a greater number of processing elements,

any of which can fail. The system must be capable of reinitializing itself to a state where the integrity of data and state of ongoing computation is preserved with only some possible performance degradation.

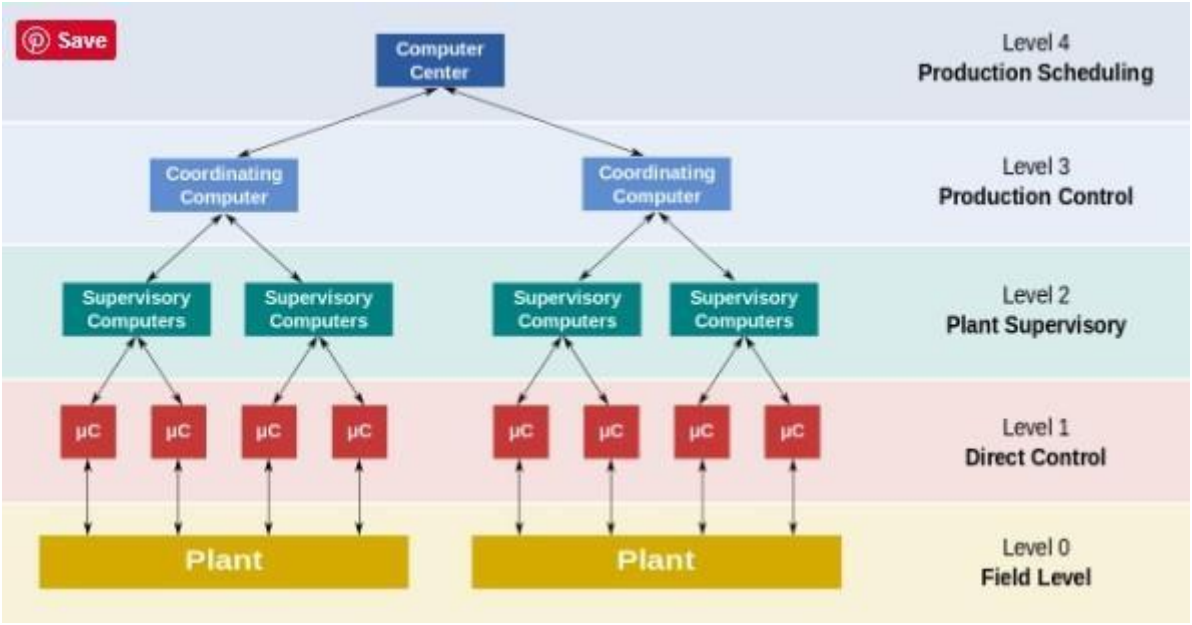


Figure 1.6 Distributed Computing architecture

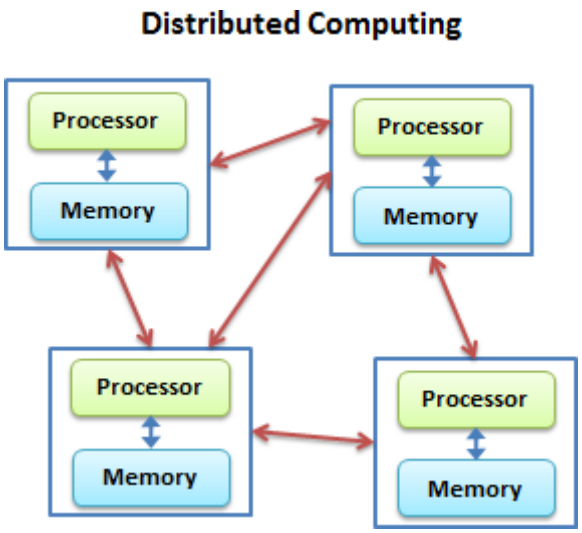
There are multiple advantages of using distributed computing. It allows scalability and makes it easier to share resources easily. It also helps to perform computation tasks efficiently. On the other hand, it is difficult to develop distributed systems. Moreover, there can be network issues.

Distributed Computing

In daily life, an individual can use a computer to work with applications such as Microsoft Word, Microsoft PowerPoint. Complex problems may not be accomplished by using a single computer. Therefore, the single problem can be divided into multiple tasks and distributed to many computers. These computers can communicate with other computers through the network. They all perform similarly to a single entity. The process of dividing a single task among multiple computers is known as distributed computing. Each computer in a distributed system is known as a node. A set of nodes is a cluster.

Distributed computing is used in many applications today. Some examples are Facebook and Google. They consist of millions and millions of users. All users communicate with others, share photographs etc. This large amount of data is stored using distributed computing. Automated teller machines in banks, telephone networks, cellular networks, distributed databases also use distributed computing.

Distributed computing provides multiple advantages. Distributed systems are extendable to the increasing growth. It provides scalability, and it is easy to share resources. Some disadvantages are there can be network issues, and it is difficult to develop distributed software.



In distributed computing systems, multiple system processors can communicate with each other using messages that are sent over the network. Such systems are increasingly available these days because of the availability at low price of computer processors and the high-bandwidth links to connect them.

The following reasons explain why a system should be built distributed, not just parallel:

Scalability: As distributed systems do not have the problems associated with shared memory, with the increased number of processors, they are obviously regarded as more scalable than parallel systems.

Reliability: The impact of the failure of any single subsystem or a computer on the network of computers defines the reliability of such a connected system. Definitely, distributed systems demonstrate a better aspect in this area compared to the parallel systems.

Data sharing: Data sharing provided by distributed systems is similar to the data sharing provided by distributed databases. Thus, multiple organizations can have distributed systems with the integrated applications for data exchange.

Resources sharing: If there exists an expensive and a special purpose resource or a processor, which cannot be dedicated to each processor in the system, such a resource can be easily shared across distributed systems.

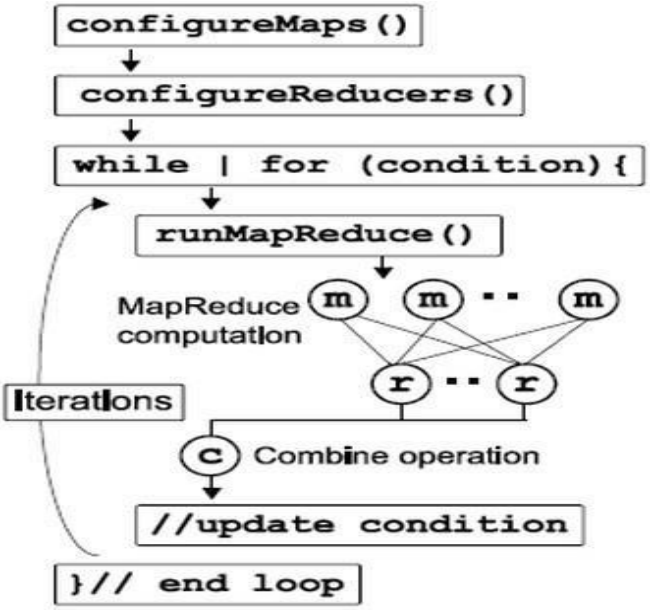
Heterogeneity and modularity: A system should be flexible enough to accept a new heterogeneous processor to be added into it and one of the processors to be replaced or removed from the system without affecting the overall system processing capability. Distributed systems are observed to be more flexible in this respect.

Geographic construction: The geographic placement of different subsystems of an application may be inherently placed as distributed. Local processing may be forced by the low communication bandwidth more specifically within a wireless network.

Economic: With the evolution of modern computers, high-bandwidth networks and workstations are available at low cost, which also favors distributed computing for economic reasons.

MapReduce

As we discussed that map reduce is really a robust framework manage large amount of data. The map reduce framework has to involve a lot of overhead when dealing with iterative map reduce.Twister is a great framework to perform iterative map reduce.



Additional functionality:

1.) **Static and variable Data :** Any iterative algorithm requires a static and variable data. Variable data are computed with static data (Usually the larger part of both) to generate another set of variable data. The process is repeated till a given condition and constrain is met. In a normal map-reduce function using Hadoop

2.) or DryadLINQ the static data are loaded uselessly every time the computation has to be performed. This is an extra overhead for the computation. Even though they remain fixed throughout the computation they have to be loaded again and again.

Twister introduces a "config" phase for both map and reduces to load any static data that is required. Loading static data for once is also helpful in running a long running Map/Reduce task

2.) **Fat Map task :** To save the access a lot of data the map is provided with an option of configurable map task, the map task can access large block of data or files. This makes it easy to add heavy computational weight on the map side.

3.) Combine operation: Unlike GFS where the output of reducer are stored in separate files, Twister comes with a new phase along with map reduce called combine that's collectively adds up the output coming from all the reducer.

4.) Programming extensions: Some of the additional functions to support iterative functionality of Twister are:
i) mapReduceBCast(Value value) for sending a single to all map tasks. For example, the "Value" can be a set of parameters, a resource (file or executable) name, or even a block of data

ii) configureMaps(Value[]values) and configureReduce(Value[]values) to configure map and reduce with additional static data

TWISTER ARCHITECTURE

The Twister is designed to effectively support iterative MapReduce function. To reach this flexibility it reads data from the local disk of the worker nodes and handle the intermediate data data in the distributed memory of the workers mode.

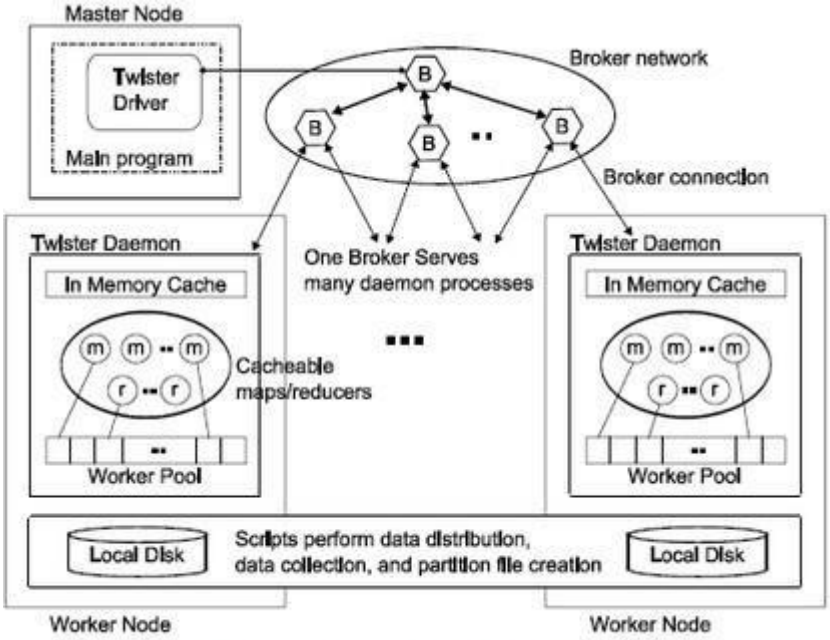


Figure 2. Architecture of Twister.

The messaging infrastructure in twister is called broker network and it is responsible to perform data transfer using publish/subscribe messaging.

Twister has three main entity:

1. Client Side Driver responsible to drive entire MapReduce computation
2. Twister Daemon running on every working node.
3. The broker Network.

Access Data

1. To access input data for map task it either reads dta from the local disk of the worker nodes.
2. receive data directly via the broker network.

They keep all data read as file and having data as native file allows Twister to pass data directly to any executable. Additionally they allow tool to perform typical file operations like

(i) create directories, (ii) delete directories, (iii) distribute input files across worker nodes, (iv) copy a set of resources/input files to all worker nodes, (v) collect output files from the worker nodes to a given location, and (vi) create partition-file for a given set of data that is distributed across the worker nodes.

Intermediate Data

The intermediate data are stored in the distributed memory of the worker node. Keeping the map output in distributed memory enhances the speed of the computation by sending the output of the map from these memory to reduces.

Messaging

The use of publish/subscribe messaging infrastructure improves the efficiency of Twister runtime. It use scalable NaradaBrokering messaging infrastructure to connect difference Broker network and reduce load on any one of them.

Fault Tolerance

There are three assumption for for providing fault tolerance for iterative mapreduce:

(i) failure of master node is rare adn no support is provided for that.

(ii) Independent of twister runtime the communication network can be made fault tolerant.

(iii) the data is replicated among the nodes of the computation infrastructure. Based on these assumptions we tryto handle failures of map/reduce tasks, daemons, and worker nodes failures.

What is the Difference between Parallel and Distributed Computing?

S.NO	PARALLEL COMPUTING	DISTRIBUTED COMPUTING
1.	Many operations are performed simultaneously	System components are located at different locations
2.	Single computer is required	Uses multiple computers
3.	Multiple processors perform multiple operations	Multiple computers perform multiple operations
4.	It may have shared or distributed memory	It have only distributed memory
5.	Processors communicate with each other through bus	Computer communicate with each other through message passing.
6.	Improves the system performance	Improves system scalability, fault tolerance and resource sharing capabilities

Parallel vs Distributed Computing	
Parallel computing is a computation type in which multiple processors execute multiple tasks simultaneously.	Distributed computing is a computation type in which networked computers communicate and coordinate the work through message passing to achieve a common goal.

Number of Computers Required	
Parallel computing occurs on one computer.	Distributed computing occurs between multiple computers.
Processing Mechanism	
In parallel computing multiple processors perform processing.	In distributed computing, computers rely on message passing.
Synchronization	
All processors share a single master clock for synchronization.	There is no global clock in distributed computing, it uses synchronization algorithms.
Memory	
In Parallel computing, computers can have shared memory or distributed memory.	In Distributed computing, each computer has their own memory.
Usage	
Parallel computing is used to increase performance and for scientific computing.	Distributed computing is used to share resources and to increase scalability.

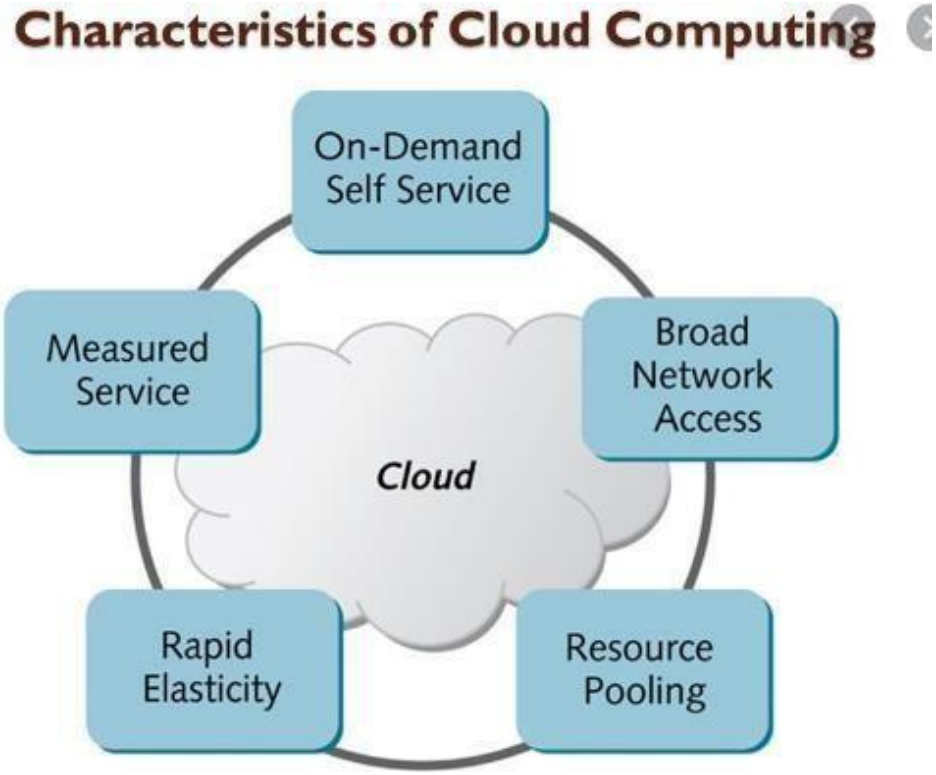
Summary – Parallel vs Distributed Computing

Parallel computing and distributed computing are two types of computation. This article discussed the difference between Parallel and Distributed Computing. The difference between parallel and distributed computing is that parallel computing is to execute multiple tasks using multiple processors simultaneously while in parallel computing, multiple computers are interconnected via a network to communicate and collaborate in order to achieve a common goal. Parallel computing is mainly used for increasing performance. Distributed computing is used to coordinate the use of shared resources or to provide communication services to theusers.

CLOUD CHARACTERISTICS

- **On Demand Self-services**
- **Broad Network Access**
- **Resource Pooling**
- **Rapid Elasticity**
- **Measured Service**
- **Dynamic Computing Infrastructure**
- **IT Service-centric Approach**
- **Minimally or Self-managed Platform**
- **Consumption-based Billing**

- **Multi Tenancy**
- **Managed Metering**



On-Demand Self-Service

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad network access

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

Resource pooling

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid elasticity

Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service.

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user

accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Large Network Access

The user can access the data of the cloud or upload the data to the cloud from anywhere just with the help of a device and an internet connection. These capabilities are available all over the network and accessed with the help of internet.

Availability

The capabilities of the Cloud can be modified as per the use and can be extended a lot. It analyzes the storage usage and allows the user to buy extra Cloud storage if needed for a very small amount.

Automatic System

Cloud computing automatically analyzes the data needed and supports a metering capability at some level of services. We can monitor, control, and report the usage. It will provide transparency for the host as well as the customer.

Economical

It is the one-time investment as the company (host) has to buy the storage and a small part of it can be provided to the many companies which save the host from monthly or yearly costs. Only the amount which is spent is on the basic maintenance and a few more expenses which are very less.

Security

Cloud Security, is one of the best features of cloud computing. It creates a snapshot of the data stored so that the data may not get lost even if one of the servers gets damaged. The data is stored within the storage devices, which cannot be hacked and utilized by any other person. The storage service is quick and reliable.

ELASTICITY IN CLOUD

Elasticity is the ability to grow or shrink infrastructure resources dynamically as needed to adapt to workload changes in an autonomic manner, maximizing the use of resources.

Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible.

Elasticity in cloud infrastructure involves enabling the hypervisor to create virtual machines or containers with the resources to meet the real-time demand. Scalability often is discussed at the application layer, highlighting capability of a system, network or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth.

This can result in savings infrastructure costs overall. Not everyone can benefit from elastic services though. Environments that do not experience sudden or cyclical changes in demand may not benefit from the cost savings elastic services offer. Use of “Elastic Services” generally implies all resources in the infrastructure be elastic. This includes but not limited to hardware, software, QoS and other policies, connectivity, and other resources that are used in elastic applications. This may become a negative trait where performance of certain applications must have guaranteed performance. It depends on the environment.

This is the case for businesses with dynamic resource demands like streaming services or e-commerce marketplaces. Various seasonal events (like Christmas, Black Friday) and other engagement triggers (like when HBO's Chernobyl spiked an interest in nuclear-related products) cause spikes of customer activity. These volatile ebbs and flows of workload require flexible resource management to handle the operation consistently.

Dimensions and Core Aspects

Any given adaptation process is defined in the context of at least one or possibly multiple types of resources that can be scaled up or down as part of the adaptation. Each resource type can be seen as a separate dimension of the adaptation process with its own elasticity properties. If a resource type is a container of other resources types, like in the case of a virtual machine having assigned CPU cores and RAM, elasticity can be considered at multiple levels. Normally, resources of a given resource type can only be provisioned in discrete units like CPU cores, virtual machines (VMs), or physical nodes. For each dimension of the adaptation process with respect to a specific resource type, elasticity captures the following core aspects of the adaptation:

Speed

The speed of scaling up is defined as the time it takes to switch from an under-provisioned state to an optimal or over-provisioned state. The speed of scaling down is defined as the time it takes to switch from an over-provisioned state to an optimal or under-provisioned state. The speed of scaling up/down does not correspond directly to the technical resource provisioning/de-provisioning time.

Precision

The precision of scaling is defined as the absolute deviation of the current amount of allocated resources from the actual resource demand. As discussed above, elasticity is always considered with respect to one or more resource types. Thus, a direct comparison between two systems in terms of elasticity is only possible if the same resource types (measured in identical units) are scaled. To evaluate the actual observable elasticity in a given scenario, as a first step, one must define the criterion based on which the amount of provisioned resources is considered to match the actual current demand needed to satisfy the system's given performance requirements. Based on such a matching criterion, specific metrics that quantify the above mentioned core aspects, can be defined to quantify the practically achieved elasticity in comparison to the hypothetical optimal elasticity. The latter corresponds to the hypothetical case where the system is scalable with respect to all considered elasticity dimensions without any upper bounds on the amount of resources that can be provisioned and where resources are provisioned and de-provisioned immediately as they are needed exactly matching the actual demand at any point in time. Optimal elasticity, as defined here, would only be limited by the resource scaling units.

Motivation.

As a subscription-oriented utility, cloud computing has gained growing attention in recent years in both research and industry and is widely considered as a promising way of managing and improving the utilization of data center resources and providing a wide range of computing services. Virtualization is a key enabling technology of cloud computing. System virtualization is able to provide abilities to access software and hardware resources from a virtual space and enables an execution platform to provide several concurrently usable and independent instances of virtual execution entities, often called virtual machines (VMs). A cloud computing platform relies on the virtualization technique to acquire more VMs to deal with workload surges or release VMsto

avoid resource over-provisioning. Such a dynamic resource provision and management feature is called elasticity. For instance, when VMs do not use all the provided resources, they can be logically resized and be migrated from a group of active servers to other servers, while the idle servers can be switched to the low-power modes (sleep or hibernate).

Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible. By dynamically optimizing the total amount of acquired resources, elasticity is used for various purposes. From the perspective of service providers, elasticity ensures better use of computing resources and more energy savings and allows multiple users to be served simultaneously. From a user's perspective, elasticity has been used to avoid inadequate provision of resources and degradation of system performance and also achieve cost reduction. Furthermore, elasticity can be used for other purposes, such as increasing the capacity of local resources. Hence, elasticity is the foundation of cloud performance and can be considered as a great advantage and a key benefit of cloud computing.

Over Provisioning and Under Provisioning

The main reason for cloud elasticity is to avoid either over provisioning or under provisioning of resources. Giving a cloud user either too much or too little data and resources will put that user at a disadvantage. If an enterprise has too many resources, they'll be paying for assets they aren't using. If they have too few resources, they can't run their processes correctly. Elastic systems can detect changes in workflows and processes in the cloud, automatically correcting resource provisioning to adjust for updated user projects.

- Modern business operations live on consistent performance and instant service availability.
- Cloud scalability and cloud elasticity handle these two business aspects in equal measure.
- Cloud scalability is an effective solution for businesses whose workload requirements are increasing slowly and predictably.
- Cloud elasticity is a cost-effective solution for the business with dynamic and unpredictable resource demands.

Notations and Preliminaries

For clarity and convenience, Notations describes the correlated variables which are used in the following sections. To elaborate the essence of cloud elasticity, we give the various states that are used in our discussion. Let denote the number of VMs in service and let be the number of requests in the system.

(1) Just-in-Need State. A cloud platform is in a just-in-need state if $i<j<3i$ is defined as the accumulated time in all just-in-need states.

(2) Over-provisioning State. A cloud platform is in an over-provisioning state if $0<j<i$. T_o is defined as the accumulated time in all over-provisioning states.

(3) Under-provisioning State. A cloud platform is in an under-provisioning state if $j>3i$. T_u is defined as the accumulated time in all under-provisioning states.

Notice that constants 1 and 3 in this paper are only for illustration purpose and can be any other values, depending on how an elastic cloud platform is managed. Different cloud users and/or applications may prefer different bounds of the hypothetical just-in-need states. The length of the interval between the upper (e.g.,) and lower (e.g.,) bounds controls the re-provisioning frequency. Narrowing down the interval leads to higher re-provision frequency for a fluctuating workload.

The just-in-need computing resource denotes a balanced state, in which the workload can be properly handled and quality of service (QoS) can be satisfactorily guaranteed. Computing resource over-provisioning, though QoS can be achieved, leads to extra but unnecessary cost to rent the cloud resources. Computing resource under-provisioning, on the other hand, delays the processing of workload and may be at the risk of breaking QoS commitment.

We present our elasticity definition for a realistic cloud platform and present mathematical foundation for elasticity evaluation. The definition of elasticity is given from a computational point of view and we develop a calculation formula for measuring elasticity value in virtualized clouds. Let T_m be the measuring time, which includes all the periods in the just-in-need, over-provisioning, and under-provisioning states; that is, $T_m=T_j+T_o+T_u$

Definition : The elasticity E of a cloud perform is the percentage of time when the platform is in just-in-need states; that is, $E=T_j/T_m=1-T_o/T_m-T_u/T_m$

Broadly defining, elasticity is the capability of delivering preconfigured and just-in-need virtual machines adaptively in a cloud platform upon the fluctuation of the computing resources required. Practically it is determined by the time needed from an under-provisioning or over-provisioning state to a balanced resource provisioning state. Definition 1 provides a mathematical definition which is easily and accurately measurable. Cloud platforms with high elasticity exhibit high adaptively, implying that they switch from an over-provisioning or an under-provisioning state to a balanced state almost in real time. Other cloud platforms take longer time to adjust and reconfigure computing resources. Although it is recognized that high elasticity can also be achieved via physical host standby, we argue that, with virtualization-enabled computing resource provisioning, elasticity can be delivered in a much easier way due to the flexibility of service migration and image template generation.

$$E = 1 - \frac{(T_o + T_u)}{T_m} = 1 - \frac{T_o}{T_m} - \frac{T_u}{T_m}, E = 1 - \frac{(T_o + T_u)}{T_m} = 1 - \frac{T_o}{T_m} - \frac{T_u}{T_m},$$

where T_m denotes the total measuring time, in which T_o is the over-provisioning time which accumulates each single period of time that the cloud platform needs to switch from an over-provisioning state to a balanced state and is the T_u under-provisioning time which accumulates each single period of time that the cloud platform needs to switch from an under-provisioning state to a corresponding balanced state.

Let P_j ,P_o ,and P_u be the accumulated probabilities of just-in-need states, over-provisioning states, and under-provisioning states, respectively. If T_m is sufficiently long, we have

$$P_j = T_j/T_m, P_o = T_o/T_m, \text{ and } P_u = T_u/T_m. \text{ Therefore, we get}$$

$$E = P_j = 1 - P_o - P_u.$$

Equation (E1) can be used when elasticity is measured by monitoring a real system. Equation (P_j) can be used when elasticity is calculated by using our CTMC model. If elasticity metrics are well defined, elasticity of cloud platforms could easily be captured, evaluated, and compared.

We would like to mention that the primary factors of elasticity, that is, the amount, frequency, and time of resource re-provisioning, are all summarized in T_o and T_u (i.e.P_o, and P_u). Elasticity can be increased by changing these factors. For example, one can maintain a list of standby or underutilized compute nodes. These nodes are prepared for the upcoming surge of workload, if there is any, to minimize the time needed to start these nodes. Such a hot standby strategy increases cloud elasticity by reducing T_u .

Relevant Properties of Clouds

We compare cloud elasticity with a few other relevant concepts, such as cloud resiliency, scalability, and efficiency.

Resiliency

The persistence of service delivery that can be trusted justifiably, when facing changes. Therefore, cloud resiliency implies the extent to which a cloud system withstands the external workload variation and under which no computing resource re-provisioning is needed and the ability to re-provision a cloud system in a timely manner. We think the latter implication defines the cloud elasticity while the former implication only exists in cloud resiliency. In our elasticity study, we will focus on the latter one.

Scalability

Elasticity is often confused with scalability in more ways than one. Scalability reflects the performance speedup when cloud resources are re-provisioned. In other words, scalability characterizes how *well* in terms of performance a new compute cluster, either larger or smaller, handles a given workload. On the other hand, elasticity explains how *fast* in terms of the re-provisioning time the compute cluster can be ready to process the workload. Cloud scalability is impacted by quite a few factors such as the compute node type and count and workload type and count. For example, Hadoop MapReduce applications typically scale much better than other single-thread applications. It can be defined in terms of scaling number of threads, processes, nodes, and even data centers. Cloud elasticity, on the other hand, is only constrained by the capability that a cloud service provider offers. Other factors that are relevant to cloud elasticity include the type and count of standby machines, computing resources that need to be re-provisioned. Different from cloud scalability, cloud elasticity does not concern workload/application type and count at all.

Efficiency

Efficiency characterizes how cloud resource can be efficiently utilized as it scales up or down. This concept is derived from speedup, a term that defines a relative performance after computing resource has been reconfigured. Elasticity is closely related to efficiency of the clouds. Efficiency is defined as the percentage of maximum performance (speedup or utilization) achievable. High cloud elasticity results in higher efficiency. However, this implication is not always true, as efficiency can be influenced by other factors independent of the system elasticity mechanisms (e.g., different implementations of the same operation). Scalability is affected by cloud efficiency. Thus, efficiency may enhance elasticity, but not sufficiency. This is due to the fact that elasticity depends on the resource types, but efficiency is not limited by resource types. For instance, with a multitenant architecture, users may exceed their resources quota. They may compete for resources or interfere each other's job executions.

Elastic Cloud Platform Modeling

To model elastic cloud platforms, we make the following assumptions.

(i) All VMs are homogeneous with the same service capability and are added/removed one at a time.λ

(ii) The user request arrivals are modeled as a Poisson process with rate .

(iii) The service time, the start-up time, and the shut-off time of each VM are governed by exponential distributions with rates α,μ , and , respectively .

(iv) Let i denote the number of virtual machines that are currently in service, and let denote the number of requests that are receiving service or in waiting.

(v) Let State v(i,j) denote the various states of a cloud platform when the virtual machine number is j and the request number is i . Let the hypothetical just-in-need state, over-provisioning state, and under-provisioning state be JIN, OP, and UP, respectively. We can set the equations of the relation between the virtual machine number and the request number as follows:

Let i denote the various states of a cloud platform when the virtual machine number is and the request number is . Let the hypothetical just-in-need state, over-provisioning state, and under-provisioning state be JIN, OP, and UP, respectively. We can set the equations of the relation between the virtual machine number and the request number as follows:

$$\text{statev}(i,j) = \begin{cases} \text{OP}, & \text{if } 0 \le j \le i; \\ \text{JIN}, & \text{if } i < j \le 3i; \\ \text{UP}, & \text{if } j > 3i. \end{cases} \text{statev}(i,j) = \begin{cases} \text{OP}, & \text{if } 0 \le j \le i; \\ \text{JIN}, & \text{if } i < j \le 3i; \\ \text{UP}, & \text{if } j > 3i. \end{cases}$$

The hypothetical just-in-need state, over-provisioning state, and under-provisioning state are listed in Table

VM number	Overprovisioning state	Just-in-need state	Underprovisioning state
1	$0 \leqslant j \leqslant 1$	$1 < j \leqslant 3$	$j > 3$
2	$0 \leqslant j \leqslant 2$	$2 < j \leqslant 6$	$j > 6$
3	$0 \leqslant j \leqslant 3$	$3 < j \leqslant 9$	$j > 9$
4	$0 \leqslant j \leqslant 4$	$4 < j \leqslant 12$	$j > 12$

For example:

Streaming Services. Netflix is dropping a new season of Mind hunter. The notification triggers a significant number of users to get on the service and watch or upload the episodes. Resource-wise, it is an activity spike that requires swift resource allocation.

E-commerce. Amazon has a Prime Day event with many special offers, sells-offs, promotions, and discounts. It attracts an immense amount of customers on the service that is doing different activities. Actions include searching for products, bidding, buying stuff, writing reviews, rating products. This

diverse activity requires a very flexible system that can allocate resources to one sector without dragging down others.

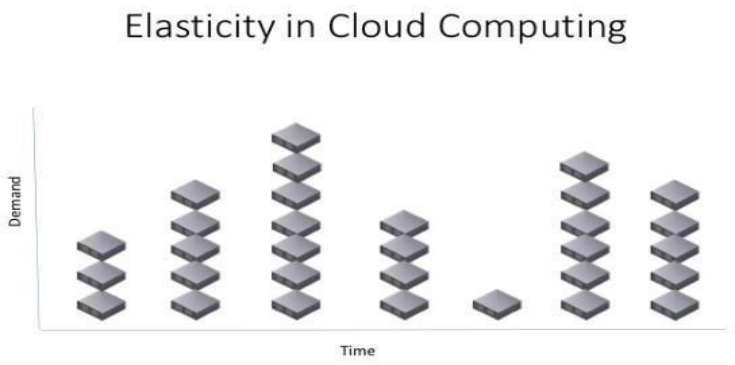


Figure 1.7 Elasticity in cloud

Cloud elasticity is a popular feature associated with scale-out solutions (horizontal scaling), which allows for resources to be dynamically added or removed when needed. Elasticity is generally associated with public cloud resources and is more commonly featured in pay-per-use or pay-as-you-grow services. This means IT managers are not paying for more resources than they are consuming and any given time. In virtualized environments cloud elasticity could include the ability to dynamically deploy new virtual machines or shutdown inactive virtual machines.

A use case that could easily have the need for cloud elasticity would be in retail with increased seasonal activity. For example, during the holiday season for black Friday spikes and special sales during this season there can be a sudden increased demand on the system. Instead of spending budget on additional permanent infrastructure capacity to handle a couple months of high load out of the year, this is a good opportunity to use an elastic solution. The additional infrastructure to handle the increased volume is only used in a pay-as-you-grow model and then "shrinks" back to a lower capacity for the rest of the year. This also allows for additional sudden and unanticipated sales activities throughout the year if needed without impacting performance or availability. This can give IT managers the security of unlimited headroom when needed. This can also be a big cost savings to retail companies looking to optimize their IT spend if packaged well by the service provider.

Elasticity as the scaling of system resources to increase or decrease capacity, whereby definitions and Specifically state that the amount of provisioned re- in a broader sense, it could also contain manual steps. Without a defined adaptation process, a scalable system cannot behave in an elastic manner, as scalability on its own does not include temporal aspects. When evaluating elasticity, the following points need to be checked beforehand:

Autonomic Scaling: What adaptation process is used for autonomic scaling?

- **Elasticity Dimensions:** What is the set of resource types scaled as part of the adaptation process?
- **Resource Scaling Units:** For each resource type, in what unit is the amount of allocated resources varied?
- **Scalability Bounds:** For each resource type, what is the upper bound on the amount of resources that can be allocated?

What is Cloud Scalability?

Scalability is one of the preeminent features of cloud computing. In the past, a system's scalability relied on the company's hardware, and thus, was severely limited in resources. With the adoption of cloud computing, scalability has become much more available and more effective.

Automatic scaling opened up numerous possibilities for the implementation of big data machine learning models and data analytics to the fold.

Overall, Cloud Scalability covers expected and predictable workload demands and also handles rapid and unpredictable changes in the scale of operation. The pay-as-you-expand pricing model makes possible the preparation of the infrastructure and its spending budget in the long term without too much strain.

There are several types of cloud scalability:

Vertical, aka Scale-Up - the ability to handle an increasing workload by adding resources to the existing infrastructure. It is a short term solution to cover immediate needs.

Horizontal, aka Scale-Out - the expansion of the existing infrastructure with new elements to tackle more significant workload requirements. It is a long term solution aimed to cover present and future resource demands with room for expansion.

Diagonal scalability is a more flexible solution that combines adding and removal of resources according to the current workload requirements. It is the most cost-effective scalability solution by far.

ON-DEMAND PROVISIONING

On-demand computing is a delivery model in which computing resources are made available to the user as needed. The resources may be maintained within the user's enterprise, or made available by a cloud service provider. When the services are provided by a third-party, the term cloud computing is often used as a synonym for on-demand computing.

The on-demand model was developed to overcome the common challenge to an enterprise of being able to meet fluctuating demands efficiently. Because an enterprise's demand on computing resources can vary drastically from one time to another, maintaining sufficient resources to meet peak requirements can be costly. Conversely, if an enterprise tried to cut costs by only maintaining minimal computing resources, it is likely there will not be sufficient resources to meet peak requirements.

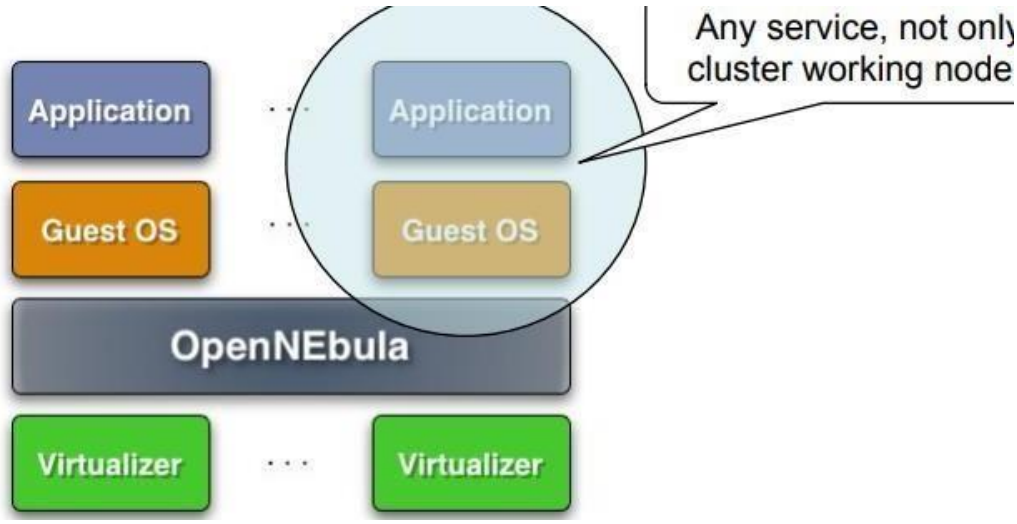
The on-demand model provides an enterprise with the ability to scale computing resources up or down with the click of a button, an API call or a business rule. The model is characterized by three attributes: scalability, pay-per-use and self-service. Whether the resource is an application program that helps team members collaborate or additional storage for archiving images, the computing resources are elastic, metered and easy to obtain.

Many on-demand computing services in the cloud are so user-friendly that non-technical end users can easily acquire computing resources without any help from the organization's information technology (IT) department. This has advantages because it can improve business agility, but it also has disadvantages because shadow IT can pose security risks. For this reason, many IT departments carry out periodic cloud audits to identify greynet on-demand applications and other rogue IT.

a) Local On-demand Resource Provisioning

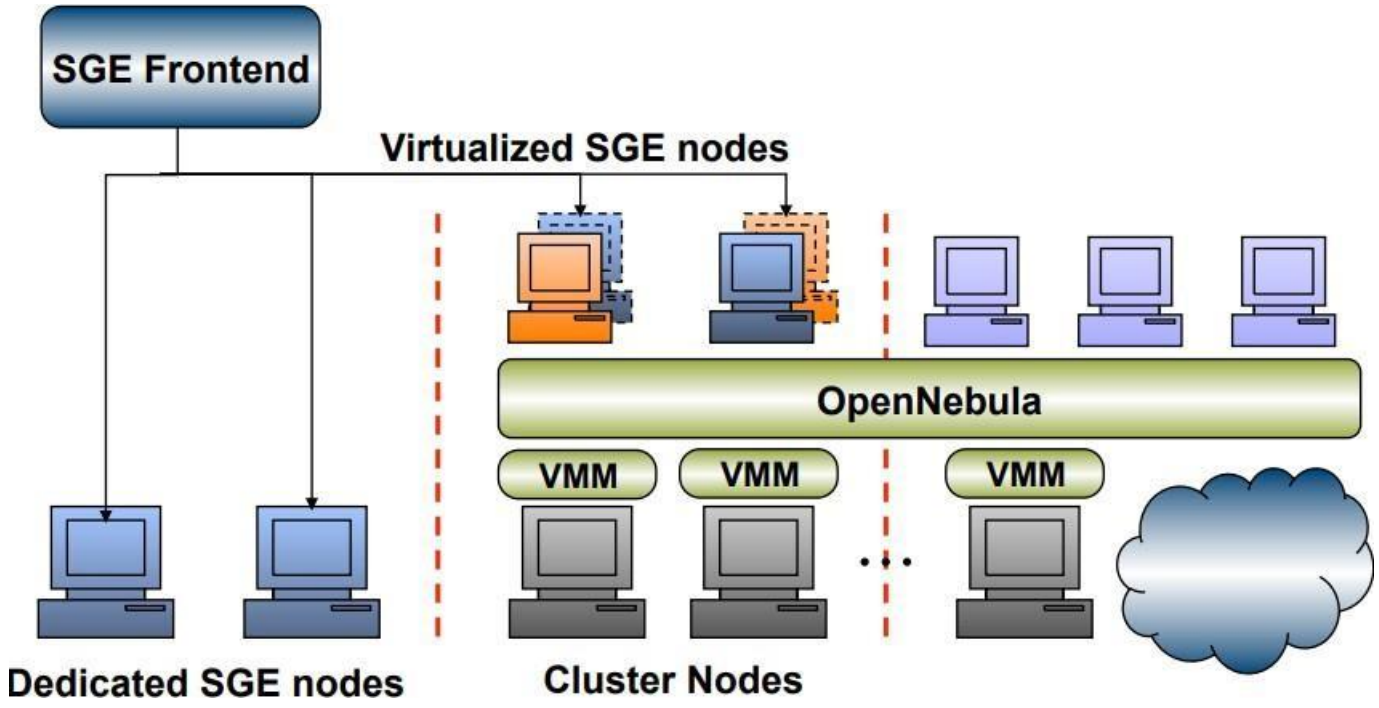
- The Engine for the Virtual Infrastructure
- Virtualization of Cluster and HPC Systems Benefits
- Open Nebula creates a distributed virtualization layer
- Extend the benefits of VM Monitors from one to multiple resources
- Decouple the VM (service) from the physical location

- Transform a distributed physical infrastructure into a flexible and elastic virtual infrastructure, which adapts to the changing demands of the VM Service workload



b)Remote On-demand Resource Provisioning

- Access to Cloud Systems
- Federation of Cloud Systems
- The RESERVOIR Project



Benefit of Remote Provisioning

The virtualization of the local infrastructure supports a virtualized alternative to contribute resources to a Grid infrastructure.

• Simpler deployment and operation of new middleware distributions

• Lower operational costs

• Easy provision of resources to more than one infrastructure or VO

• Easy support for VO-specific worker nodes

• Performance partitioning between local and grid clusters

VMs to Provide pre-Created Software Environments for Jobs

• Extensions of job execution managers to create per-job basis VMs so as to provide a pre-defined environment for job execution

• Those approaches still manage jobs

• The VMs are bounded to a given PM and only exist during job execution

• Condor, SGE, MOAB, Globus GridWay…

• Job Execution Managers for the Management of VMs

• Job execution managers enhanced to allow submission of VMs

• Those approaches manage VMs as jobs

• Condor, "pilot" backend in Globus VWS

Differences between VMs and Jobs as basic Management Entities

• VM structure: Images with fixed and variable parts for migration…

• VM life-cycle: Fixed and transient states for contextualization, live migration.

• VM duration: Long time periods ("forever")

• VM groups (services): Deploy ordering, affinity, rollback management…

• VM elasticity: Changing of capacity requirements and number of VMs

• Different Metrics in the Allocation of Physical Resources

Capacity provisioning:

Probability of SLA violation for a given cost of provisioning including support for server consolidation, partitioning.

• HPC scheduling: Turnaround time, wait time, throughput

VMware DRS, Platform Orchestrator, IBM Director, Novell ZENworks,Enomalism, Xenoserver

• Advantages:

• Open-source (Apache license v2.0)

• Open and flexible architecture to integrate new virtualization technologies

• Support for the definition of any scheduling policy (consolidation, workload balance, affinity, SLA…)

• LRM-like CLI and API for the integration of third-party tools

Static Provisioning: For applications that have predictable and generally unchanging demands/workloads, it is possible to use "static provisioning" effectively. With advance provisioning, the customer contracts with the provider for services and the provider prepares the appropriate resources in advance of start of service. The customer is charged a flat fee or is billed on a monthly basis.

Dynamic Provisioning: In cases where demand by applications may change or vary, "dynamic provisioning" techniques have been suggested whereby VMs may be migrated on-the-fly to new compute nodes within the cloud. With dynamic provisioning, the provider allocates more resources as they are needed and removes them when they are not. The customer is billed on a pay-per-use basis. When dynamic provisioning is used to create a hybrid cloud, it is sometimes referred to as cloud bursting.

User Self-provisioning: With user self- provisioning (also known as cloud self- service), the customer purchases resources from the cloud provider through a web form, creating a customer account and paying for resources with a credit card. The provider's resources are available for customer use within hours, if not minutes. **Parameters for Resource Provisioning**

i) Response time: The resource provisioning algorithm designed must take minimal time to respond when executing the task.

ii) Minimize Cost: From the Cloud user point of view cost should be minimized.

iii) Revenue Maximization: This is to be achieved from the Cloud Service Provider's view.

iv) Fault tolerant: The algorithm should continue to provide service in spite of failure ofnodes.

v) Reduced SLA Violation: The algorithm designed must be able to reduce SLAviolation.

vi) Reduced Power Consumption: VM placement & migration techniques must lower power consumption.

Resource Provisioning Strategies For efficiently making use of the Cloud Resources, resource provisioning techniques are to be used.

There are many resource provisioning techniques both static and dynamic provisioning each having its own pros and cons. The provisioning techniques are used to improve QoS parameters , minimize cost for cloud user and maximize revenue for the Cloud Service Provider improve response time , deliver services to cloud user even in presence of failures, improve performance reduces SLA violation, efficiently uses cloud resources reduces power consumption

c) Static Resource Provisioning Techniques

Aneka's deadline driven provisioning technique is used for scientific application as scientific applications require large computing power. Aneka is a cloud application platform which is capable of provisioning resources which are obtained from various sources such as public and private clouds, clusters, grids and desktop grids. This technique efficiently allocates resources thereby reducing application execution time. Because resource failures are inevitable it is a good idea to efficiently couple private and public cloud using an architectural framework for realizing the full potential of hybrid clouds. Proposes a failure- aware resource provisioning algorithm that is capable of providing cloud users' QoS requirements. This provides resource provisioning policies and proposes a scalable hybrid infrastructure to assure QoS of the users. This improves the deadline violation rate by 32% and 57% improvement in slowdown with a limited cost on a public cloud. Since resources held by single cloud are usually limited it is better to get resources from other participating clouds. But this is difficult to provide right resources from different cloud providers because management policies are different and description about various resources is different in each organization. Also interoperability is hard to achieve. To overcome this, Inter Cloud Resource Provisioning (ICRP) system is proposed in where resources and tasks are described semantically and stored using resource ontology and using a semantic scheduler and a set of inference rules resources are assigned. With the increasing functionality and complexity in Cloud computing, resource failure cannot be avoided. So the proposed strategy in addresses the question of provisioning resources to applications in the presence of failures in a hybrid cloud computing environment. It takes into account the workload model and the failure correlations to redirect requests to appropriate cloud providers. This is done using real failure traces and workload models, and it is found that the deadline violation rate of users' request is reduced by 20% with a limited cost on Amazon Public Cloud.

d) Dynamic Resource provisioning Techniques

The algorithm proposed in suitable for web applications where response time is one of the important factors. For web applications guaranteeing average response time is difficult because traffic patterns are highly dynamic and difficult to predict accurately and also due to the complex nature of the multi-tier web applications it is difficult to identify bottlenecks and resolving them automatically. This provisioning technique proposes a working prototype system for automatic detection and resolution of bottlenecks in a multi-tier cloud hosted web applications. This improves response time and also identifies over provisioned resources. VM based resource management is a heavy weight task. So this is less flexible and less resource efficient. To overcome this, a lightweight approach called Elastic Application Container [EAC] is used for provisioning the resources where EAC is a virtual resource unit for providing better resource efficiency and more scalable applications. This EAC–oriented platform and algorithm is to support multi tenant cloud use . Dynamic creation of the tenant is done by integrating cloud based services on the fly. But dynamic creation is by building the required components from the scratch. Even though multitenant systems save cost, but incur huge reconfiguration costs. This approach allows clients to specify their requirements which are not supported in previous techniques. This approach proposes a novel user interface-tenant selector (UTC) model which enables cloud based services to be systematically modeled and provisioned as variants of existing service tenants in the cloud. This considers functional, non functional and resource allocation requirements which are explicitly specified by the client via the user interface component of the model. So the cost and time is saved in this approach. The technique proposed in makes use of the provisioned called adaptive power-aware virtual machine provisioned (APA-VMP) where the resources are provisioned dynamically from the resource pool. This is from Infrastructure-as-a- Service provider point of view where the custom Virtual machines (VM) are launched in appropriate server in a data center.

Server Consolidation is a technique to save on energy costs in virtualized data centers. The instantiation of a given set of Virtual Machines (VMs) to Physical Machines (PMs) can be thought of as a provisioning step where amount of resources to be allocated to a VM is determined and a placement step which decides which VMs can be placed together on physical machines thereby allocating VMs to PMs. Here a provisioning scheme is proposed which takes into account acceptable intensity of violation of provisioned resources. Correlation among aggregated resource demands of VMs is considered when VMs are mapped to PMs. This reduces number of servers (up to 32%) required to host 1000 VMs and thus enables to turn off unnecessary servers. In Cloud Computing federated Cloud Environment is used when the resource requirement of user requests exceeds the resource limits of Cloud Providers' resources. It is desirable to reduce SLA violation which can be achieved through load balancing algorithm that is threshold based. This algorithm allocates VMs in order to balance the load among multiple datacenters in a federated cloud environment by focusing on reducing users' SLA violation.

Sl. No.	On demand Provisioning Techniques	Merits	Challenges
1	Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka	Able to efficiently allocate resources from different sources in order to reduce application execution times.	Not suitable for HPC-data intensive applications.
2	Dynamic provisioning in multi-tenant service clouds	Matches tenant functionalities with client requirements.	Does not work for testing on real-life cloud–based system and across several domains.

3	Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning	Outperforms in terms of Flexibility and resource efficiency.	Not suitable for web applications and supports only one type of programming language, Java.
4	Hybrid Cloud Resource Provisioning Policy in the Presence of Resource Failures	Able to adopt user the workload model to provide flexibility in the choice of strategy based on the desired level of QoS, the needed performance, and the available budget.	Not suitable to run real experiments.
5	Provisioning of Requests for Virtual Machine Sets with Placement Constraints in IaaS Clouds	Runtime efficient & can provide an effective means of online VM-to-PM mapping and also Maximizes revenue.	Not practical for medium to large problems.
6	Failure-aware resource provisioning for hybrid Cloud infrastructure	Able to improve the users' QoS about 32% in terms of deadline violation rate and 57% in terms of slowdown with a limited cost on a public cloud.	Not able to run real experiments and also not able to move VMs between public and private clouds to deal with resource failures in the local infrastructures.
7	VM Provisioning Method to Improve the Profit and SLA Violation of Cloud Service Providers	Reduces SLA violations & Improves Profit.	Increases the problem of resource allocation and load balancing among the datacenters.
8	Risk Aware Provisioning and Resource Aggregation based Consolidation of Virtual Machines	Significant amount of reduction in the numbers required to host 1000 VMs and enables to turn off unnecessary servers.	Takes into account only CPU requirements of VMs.
9	Semantic based Resource Provisioning and Scheduling in Inter-cloud Environment	Enables the fulfillment of customer requirements to the maximum by providing additional resources to the cloud system participating in a federated cloud environment thereby solving the interoperability problem.	QoS parameters like response time and throughput has to be achieved for interactive applications.
10	Design and implementation of adaptive power-aware virtual machine provisioned (APA-VMP) using swarm intelligence	Efficient VM placement and significant reduction in power.	Not suitable for conserving power in modern data centers.
11	Adaptive resource provisioning for read intensive multi-tier applications in the cloud	Automatic Identification and resolution of bottlenecks in multitier web application hosted on a cloud.	Not suitable for n-tier clustered application hosted on a cloud.
12	Optimal Resource Provisioning for Cloud Computing Environment	Efficiently provisions Cloud Resources for SaaS users with a limited budget and Deadline thereby optimizing QoS.	Applicable only for SaaS users and SaaS providers.

UNIT V
CLOUD TECHNOLOGIES AND ADVANCEMENTS

Hadoop

Apache Hadoop is an open source software framework used to develop data processing applications which are executed in a distributed computing environment. Applications built using HADOOP are run on large data sets distributed across clusters of commodity computers. Commodity computers are cheap and widely available. These are mainly useful for achieving greater computational power at low cost.

Similar to data residing in a local file system of a personal computer system, in Hadoop, data resides in a distributed file system which is called as a Hadoop Distributed File system. The processing model is based on 'Data Locality' concept wherein computational logic is sent to cluster nodes (server) containing data. This computational logic is nothing, but a compiled version of a program written in a high-level language such as Java. Such a program, processes data stored in Hadoop HDFS.

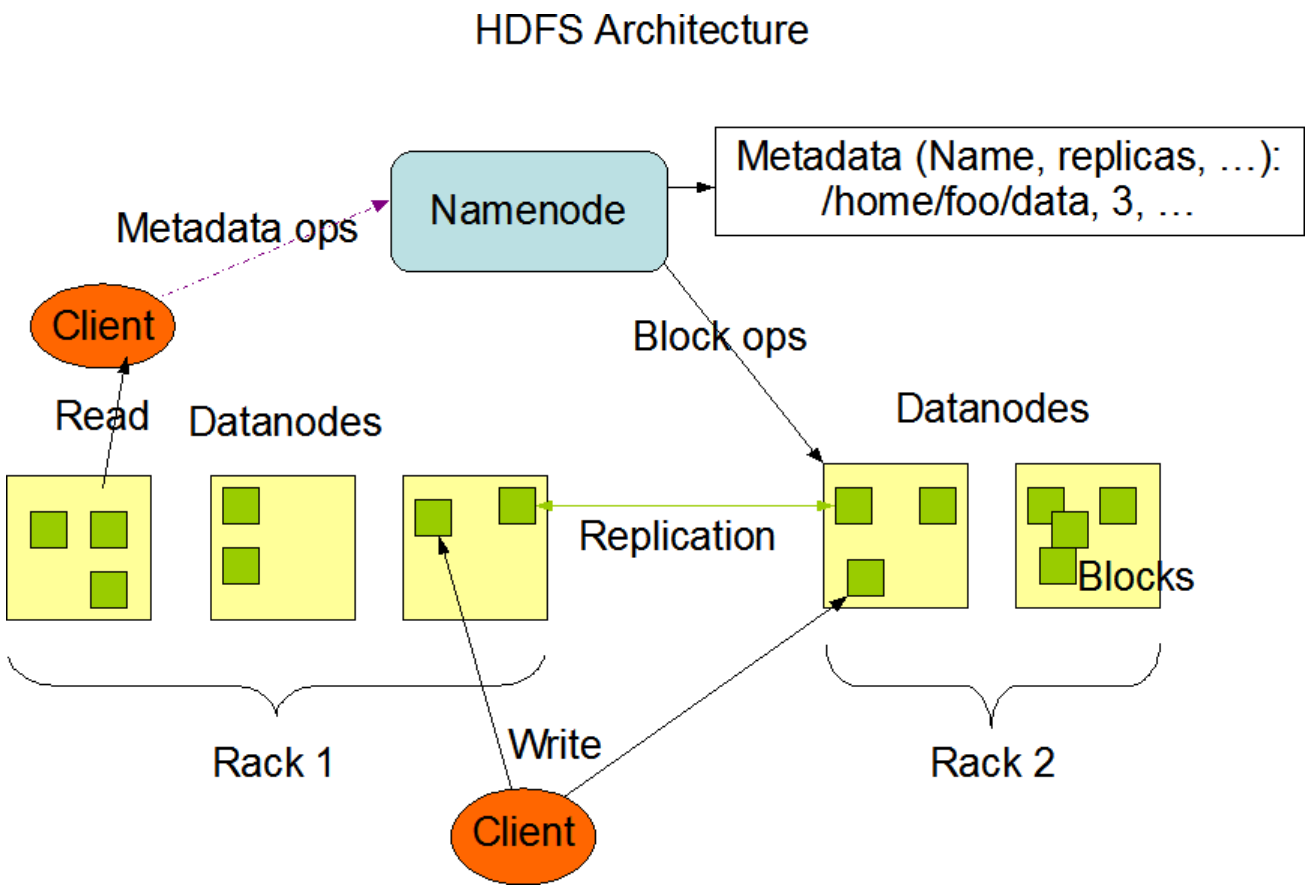
- Hadoop EcoSystem and Components
- Hadoop Architecture
- Features Of 'Hadoop'
- Network Topology In Hadoop

Apache Hadoop consists of two sub-projects –

Hadoop MapReduce: MapReduce is a computational model and software framework for writing applications which are run on Hadoop. These MapReduce programs are capable of processing enormous data in parallel on large clusters of computation nodes.

HDFS (Hadoop Distributed File System): HDFS takes care of the storage part of Hadoop applications. MapReduce applications consume data from HDFS. HDFS creates multiple replicas of data blocks and distributes them on compute nodes in a cluster. This distribution enables reliable and extremely rapid computations.

Although Hadoop is best known for MapReduce and its distributed file system- HDFS, the term is also used for a family of related projects that fall under the umbrella of distributed computing and large-scale data processing.



NameNode and DataNodes

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case.

The existence of a single NameNode in a cluster greatly simplifies the architecture of the system. The NameNode is the arbitrator and repository for all HDFS metadata. The system is designed in such a way that user data never flows through the NameNode.

The File System Namespace

HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file. HDFS supports user quotas and access permissions. HDFS does not support hard links or soft links. However, the HDFS architecture does not preclude implementing these features.

While HDFS follows naming convention of the FileSystem, some paths and names (e.g. /.reserved and .snapshot) are reserved. Features such as transparent encryption and snapshot use reserved paths.

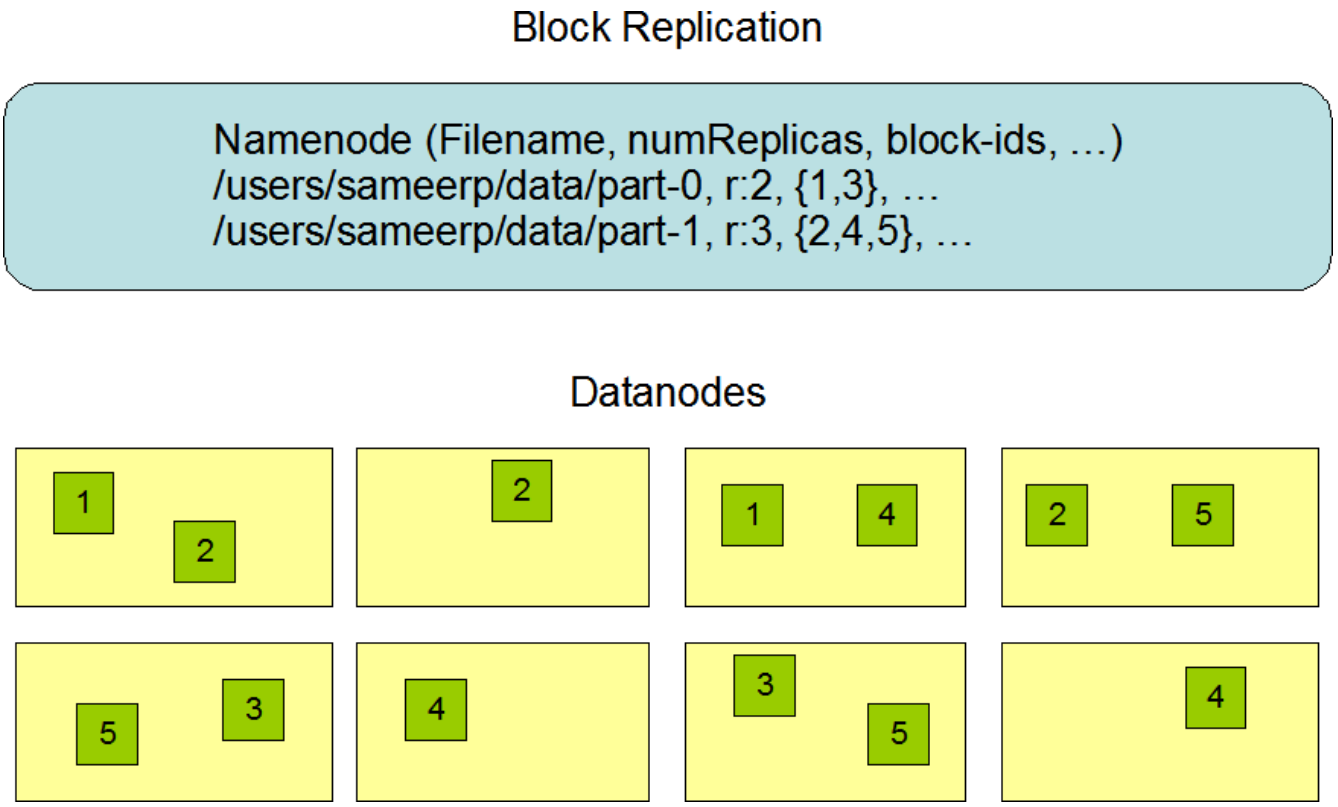
The NameNode maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the NameNode. An application can specify the number of replicas of a file that should be maintained by HDFS. The number of copies of a file is called the replication factor of that file. This information is stored by the NameNode.

Data Replication

HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. All blocks in a file except the last block are the same size, while users can start a new block without filling out the last block to the configured block size after the support for variable length block was added to append and hsync. An application can specify the number of replicas of a file.

The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once (except for appends and truncates) and have strictly one writer at any time.

The NameNode makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode.



Replication

The placement of replicas is critical to HDFS reliability and performance. Optimizing replica placement distinguishes HDFS from most other distributed file systems. This is a feature that needs lots of tuning and experience. The purpose of a rack-aware replica placement policy is to improve data reliability, availability, and network bandwidth utilization. The current implementation for the replica placement policy is a first effort in this direction. The short-term goals of implementing this policy are to validate it on production systems, learn more about its behavior, and build a foundation to test and research more sophisticated policies.

Large HDFS instances run on a cluster of computers that commonly spread across many racks. Communication between two nodes in different racks has to go through switches. In most cases, network bandwidth between machines in the same rack is greater than network bandwidth between machines in different racks. The NameNode determines the rack id each DataNode belongs to via the process outlined in Hadoop Rack Awareness. A simple but non-optimal policy is to place replicas on unique racks. This prevents losing data when an entire rack fails and allows use of bandwidth from multiple racks when reading data. This policy evenly distributes replicas in the cluster which makes it easy to balance load on component failure. However, this policy increases the cost of writes because a write needs to transfer blocks to multiple racks.

For the common case, when the replication factor is three, HDFS's placement policy is to put one replica on the local machine if the writer is on a datanode, otherwise on a random datanode in the same

rack as that of the writer, another replica on a node in a different (remote) rack, and the last on a different node in the same remote rack. This policy cuts the inter-rack write traffic which generally improves write performance. The chance of rack failure is far less than that of node failure; this policy does not impact data reliability and availability guarantees. However, it does reduce the aggregate network bandwidth used when reading data since a block is placed in only two unique racks rather than three. With this policy, the replicas of a file do not evenly distribute across the racks. One third of replicas are on one node, two thirds of replicas are on one rack, and the other third are evenly distributed across the remaining racks. This policy improves write performance without compromising data reliability or read performance.

If the replication factor is greater than 3, the placement of the 4th and following replicas are determined randomly while keeping the number of replicas per rack below the upper limit (which is basically (replicas - 1) / racks + 2).Because the NameNode does not allow DataNodes to have multiple replicas of the same block, maximum number of replicas created is the total number of DataNodes at that time.

After the support for Storage Types and Storage Policies was added to HDFS, the NameNode takes the policy into account for replica placement in addition to the rack awareness described above. The NameNode chooses nodes based on rack awareness at first, then checks that the candidate node have storage required by the policy associated with the file. If the candidate node does not have the storage type, the NameNode looks for another node. If enough nodes to place replicas cannot be found in the first path, the NameNode looks for nodes having fallback storage types in the second path.

Replica Selection

To minimize global bandwidth consumption and read latency, HDFS tries to satisfy a read request from a replica that is closest to the reader. If there exists a replica on the same rack as the reader node, then that replica is preferred to satisfy the read request. If HDFS cluster spans multiple data centers, then a replica that is resident in the local data center is preferred over any remote replica.

MapReduce

What is MapReduce?

Hadoop MapReduce (Hadoop Map/Reduce) is a software framework for distributed processing of large data sets on computing clusters. It is a sub-project of the Apache Hadoop project. Apache Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. MapReduce is the core component for data processing in Hadoop framework. In layman’s term Mapreduce helps to split the input data set into a number of parts and run a program on all data parts parallel at once. The term MapReduce refers to two

separate and distinct tasks. The first is the map operation, takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce operation combines those data tuples based on the key and accordingly modifies the value of the key.

Map Task

The Map task run in the following phases:-

a. RecordReader

The recordreader transforms the input split into records. It parses the data into records but does not parse records itself. It provides the data to the mapper function in key-value pairs. Usually, the key is the positional information and value is the data that comprises the record.

b. Map

In this phase, the mapper which is the user-defined function processes the key-value pair from the recordreader. It produces zero or multiple intermediate key-value pairs. The decision of what will be the key-value pair lies on the mapper function. The key is usually the data on which the reducer function does the grouping operation. And value is the data which gets aggregated to get the final result in the reducer function.

c. Combiner

The combiner is actually a localized reducer which groups the data in the map phase. It is optional. Combiner takes the intermediate data from the mapper and aggregates them. It does so within the small scope of one mapper. In many situations, this decreases the amount of data needed to move over the network. For example, moving (Hello World, 1) three times consumes more network bandwidth than moving (Hello World, 3). Combiner provides extreme performance gain with no drawbacks. The combiner is not guaranteed to execute. Hence it is not of overall algorithm.

d. Partitioner

Partitioner pulls the intermediate key-value pairs from the mapper. It splits them into shards, one shard per reducer. By default, partitioner fetches the hashcode of the key. The partitioner performs modulus operation by a number of reducers: key.hashcode()%(number of reducers). This distributes the keyspace evenly over the reducers. It also ensures that key with the same value but from different mappers end up into the same reducer. The partitioned data gets written on the local file system from each map task. It waits there so that reducer can pull it.

b. Reduce Task

The various phases in reduce task are as follows:

i. Shuffle and Sort

The reducer starts with shuffle and sort step. This step downloads the data written by partitioner to the machine where reducer is running. This step sorts the individual data pieces into a large data list. The purpose of this sort is to collect the equivalent keys together. The framework does this so that we could iterate over it easily in the reduce task. This phase is not customizable. The framework handles everything

automatically. However, the developer has control over how the keys get sorted and grouped through a comparator object.

ii. Reduce

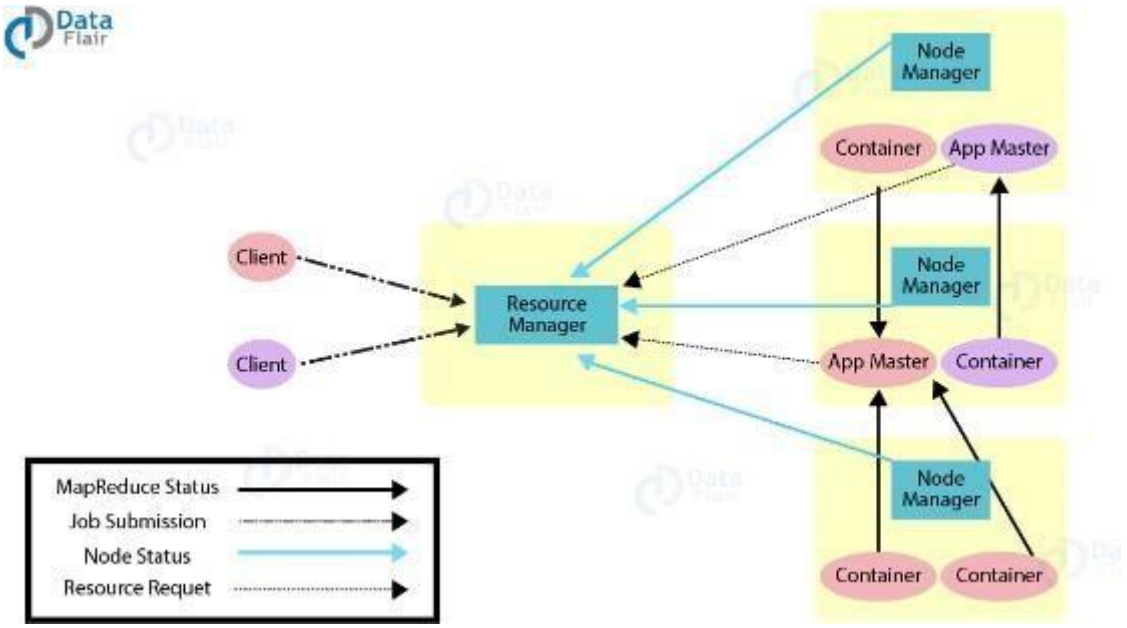
The reducer performs the reduce function once per key grouping. The framework passes the function key and an iterator object containing all the values pertaining to the key. We can write reducer to filter, aggregate and combine data in a number of different ways. Once the reduce function gets finished it gives zero or more key-value pairs to the output format. Like map function, reduce function changes from job to job. As it is the core logic of the solution.

iii. Output Format

This is the final step. It takes the key-value pair from the reducer and writes it to the file by record writer. By default, it separates the key and value by a tab and each record by a newline character. We can customize it to provide richer output format. But none the less final data gets written to HDFS.

YARN

YARN or Yet Another Resource Negotiator is the resource management layer of Hadoop. The basic principle behind YARN is to separate resource management and job scheduling/monitoring function into separate daemons. In YARN there is one global ResourceManager and per-application ApplicationMaster. An Application can be a single job or a DAG of jobs.Inside the YARN framework, we have two daemons ResourceManager and NodeManager. The ResourceManager arbitrates resources among all the competing applications in the system. The job of NodeManger is to monitor the resource usage by the container and report the same to ResourceManger. The resources are like CPU, memory, disk, network and so on.



i. Scheduler

Scheduler is responsible for allocating resources to various applications. This is a pure scheduler as it does not perform tracking of status for the application. It also does not reschedule the tasks which fail due

to software or hardware errors. The scheduler allocates the resources based on the requirements of the applications.

ii. Application Manager

Following are the functions of ApplicationManager

- Accepts job submission.
- Negotiates the first container for executing ApplicationMaster. A container incorporates elements such as CPU, memory, disk, and network.
- Restarts the ApplicationMaster container on failure.

Functions of ApplicationMaster:-

- Negotiates resource container from Scheduler.
- Tracks the resource container status.
- Monitors progress of the application.

We can scale the YARN beyond a few thousand nodes through YARN Federation feature. This feature enables us to tie multiple YARN clusters into a single massive cluster. This allows for using independent clusters, clubbed together for a very large job.

iii. Features of Yarn

YARN has the following features:-

a. Multi-tenancy

YARN allows a variety of access engines (open-source or propriety) on the same Hadoop data set. These access engines can be of batch processing, real-time processing, iterative processing and so on.

b. Cluster Utilization

With the dynamic allocation of resources, YARN allows for good use of the cluster. As compared to static map-reduce rules in previous versions of Hadoop which provides lesser utilization of the cluster.

c. Scalability

Any data center processing power keeps on expanding. YARN's ResourceManager focuses on scheduling and copes with the ever-expanding cluster, processing petabytes of data.

d. Compatibility

MapReduce program developed for Hadoop 1.x can still on this YARN. And this is without any disruption to processes that already work.

Virtual Box

VirtualBox is opensource software for virtualizing the X86 computing architecture. It acts as a hypervisor, creating a VM (Virtual Machine) in which the user can run another OS (operating system). The operating system in which VirtualBox runs is called the "host" OS. The operating system running in the VM is called the "guest" OS. VirtualBox supports Windows, Linux, or macOS as its host OS. When configuring a virtual machine, the user can specify how many CPU cores, and how much RAM and disk

space should be devoted to the VM. When the VM is running, it can be "paused." System execution is frozen at that moment in time, and the user can resume using it later.

Why Is VirtualBox Useful?

One:

VirtualBox allows you to run more than one operating system at a time. This way, you can run software written for one operating system on another (for example, Windows software on Linux or a Mac) without having to reboot to use it (as would be needed if you used partitioning and dual-booting). You can also configure what kinds of "virtual" hardware should be presented to each such operating system, and you can install an old operating system such as DOS or OS/2 even if your real computer's hardware is no longer supported by that operating system.

Two:

Sometimes, you may want to try out some new software, but would rather not chance it mucking up the pretty decent system you've got right now. Once installed, a virtual machine and its virtual hard disks can be considered a "container" that can be arbitrarily frozen, woken up, copied, backed up, and transported between hosts.

By using a VirtualBox feature called "snapshots", you can save a particular state of a virtual machine and revert back to that state, if necessary. This way, you can freely experiment with a computing environment. If something goes wrong (e.g. after installing misbehaving software or infecting the guest with a virus), you can easily switch back to a previous snapshot and avoid the need of frequent backups and restores.

Three:

Software vendors can use virtual machines to ship entire software configurations. For example, installing a complete mail server solution on a real machine can be a tedious task (think of rocket science!). With VirtualBox, such a complex setup (then often called an "appliance") can be packed into a virtual machine. Installing and running a mail server becomes as easy as importing such an appliance into VirtualBox.

Along these same lines, I find the "clone" feature of virtual box just awesome! By cloning virtual machines, I'm able to move them from one machine to another along with all saved snapshots. If you try to imagine what it would involve to do something similar with physical machines, you will immediately see the power of this feature. Do have a look at my tutorial on moving virtual machines with snapshots.

Four:

On an enterprise level, virtualization can significantly reduce hardware and electricity costs. Most of the time, computers today only use a fraction of their potential power and run with low average system loads. A lot of hardware resources as well as electricity is thereby wasted. So, instead of running many such physical computers that are only partially used, one can pack many virtual machines onto a few powerful hosts and balance the loads between them.

VirtualBox Terminology

- When dealing with virtualization, it helps to acquaint oneself with a bit of crucial terminology, especially the following terms:

Host Operating System (Host OS):

- The operating system of the physical computer on which VirtualBox was installed. There are versions of VirtualBox for Windows, Mac OS X, Linux and Solaris hosts.

Guest Operating System (Guest OS):

- The operating system that is running inside the virtual machine.

Virtual Machine (VM):

We've used this term often already. It is the special environment that VirtualBox creates for your guest operating system while it is running. In other words, you run your guest operating system "in" a VM. Normally, a VM will be shown as a window on your computers desktop, but depending on which of the various frontends of VirtualBox you use, it can be displayed in full screen mode or remotely on another computer.

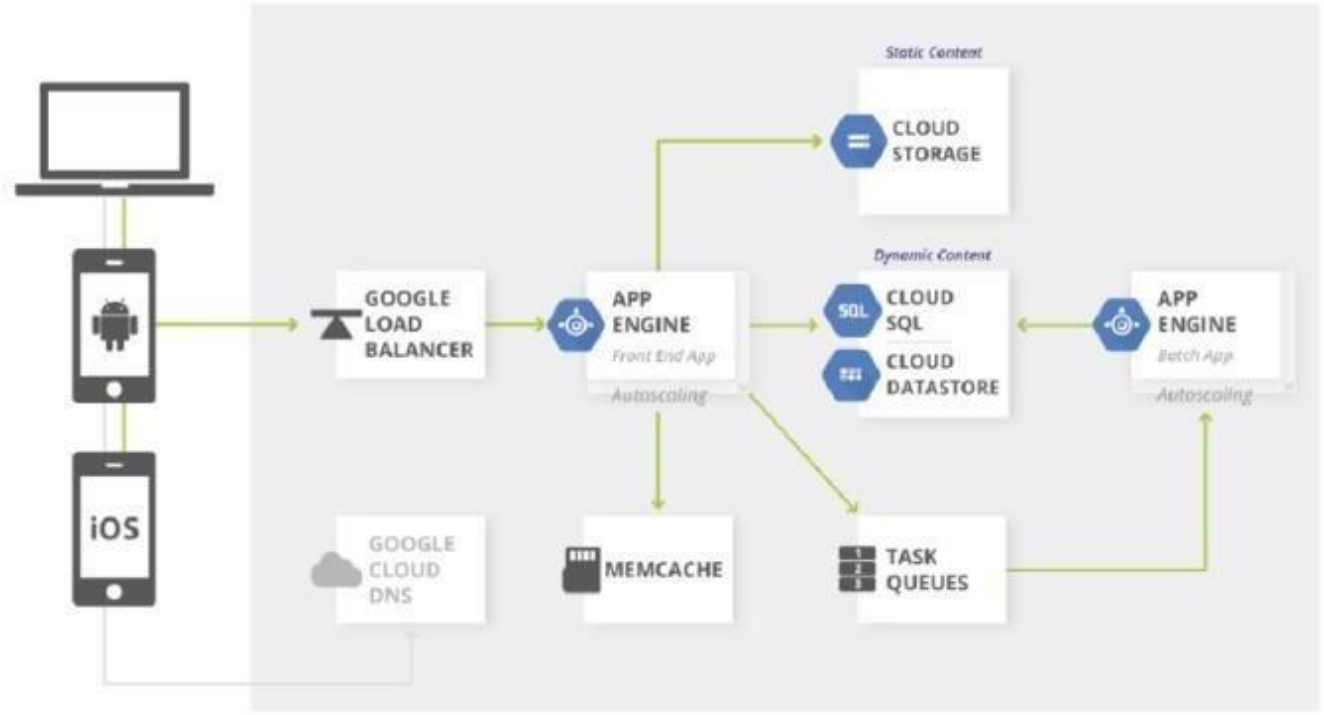
Google App Engine

App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. You can choose from several popular languages, libraries, and frameworks to develop your apps, then let App Engine take care of provisioning servers and scaling your app instances based on demand

- The App Engine requires that apps be written in Java or Python, store data in Google BigTable and use the Google query language. Non-compliant applications require modification to use App Engine.

- Google App Engine provides more infrastructure than other scalable hosting services such as Amazon Elastic Compute Cloud (EC2). The App Engine also eliminates some system administration and developmental tasks to make it easier to write scalable applications.

- Google App Engine is free up to a certain amount of resource usage. Users exceeding the per-day or per-minute usage rates for CPU resources, storage, number of API calls or requests and concurrent requests can pay for more of these resources.



Modern web applications

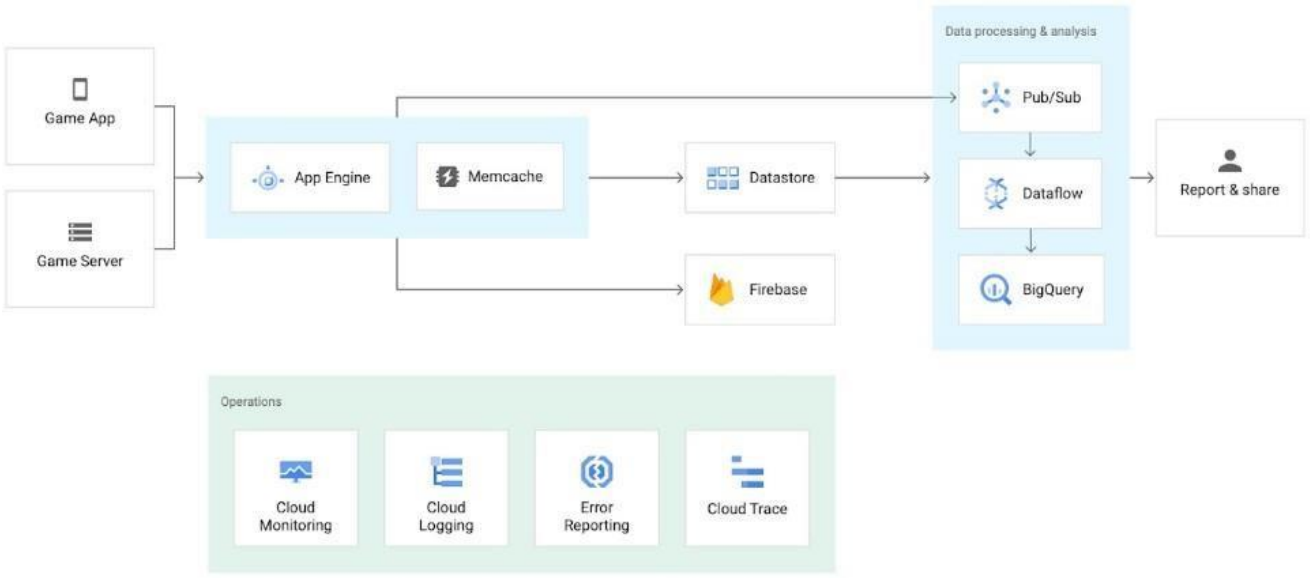
Quickly reach customers and end users by deploying web apps on App Engine. With zero-config deployments and zero server management, App Engine allows you to focus on writing code. Plus, App Engine automatically scales to support sudden traffic spikes without provisioning, patching, or monitoring.

Below is a sample reference architecture for building a simple web app using App Engine and Google Cloud.

Scalable mobile back ends

Whether you're building your first mobile app or looking to reach existing users via a mobile experience, App Engine automatically scales the hosting environment for you. Plus, seamless integration with Firebase provides an easy-to-use frontend mobile platform along with the scalable and reliable backend.

Below is sample reference architecture for a typical mobile app built using both Firebase and App Engine along with other services in Google Cloud.



Features

- Popular languages
- Build your application in Node.js, Java, Ruby, C#, Go, Python, or PHP—or bring your own language runtime.

Open and flexible

- Custom runtimes allow you to bring any library and framework to App Engine by supplying a Docker container.

Fully managed

- A fully managed environment lets you focus on code while App Engine manages infrastructure concerns.

Powerful application diagnostics

- Use Cloud Monitoring and Cloud Logging to monitor the health and performance of your app and Cloud Debugger and Error Reporting to diagnose and fix bugs quickly.

Application versioning

- Easily host different versions of your app, easily create development, test, staging, and production environments.

Traffic splitting

- Route incoming requests to different app versions, A/B test, and do incremental feature rollouts.

Application security

- Help safeguard your application by defining access rules with App Engine firewall and leverage managed SSL/TLS certificates* by default on your custom domain at no additional cost.

Services ecosystem

- Tap a growing ecosystem of Google Cloud services from your app including an excellent suite of cloud developer tools.

Advantages of Google App Engine

There are many advantages to the Google App Engine that helps to take your app ideas to the next level. This includes:

Infrastructure for Security

- Around the world, the Internet infrastructure that Google has is probably the most secure. There is rarely any type of unauthorized access to date as the application data and code are stored in highly secure servers.

- You can be sure that your app will be available to users worldwide at all times since Google has several hundred servers globally. Google's security and privacy policies are applicable to the apps developed using Google's infrastructure

Quick to Start

With no product or hardware to purchase and maintain, you can prototype and deploy the app to your users without taking much time.

Easy to Use

Google App Engine (GAE) incorporates the tools that you need to develop, test, launch, and update the applications.

Scalability

- For any app's success, this is among the deciding factors. Google creates its own apps using GFS, Big Table and other such technologies, which are available to you when you utilize the Google app engine to create apps.
- You only have to write the code for the app and Google looks after the testing on account of the automatic scaling feature that the app engine has. Regardless of the amount of data or number of users that your app stores, the app engine can meet your needs by scaling up or down as required.
  - The good thing about Google App Engine as a manageable platform is that it has made it feasible for our engineers to effortlessly scale up their applications with no-operations skill. It, additionally, sets us up with the best practices as far as logging, security and releasing management is concerned.

Performance and Reliability

Google is among the leaders worldwide among global brands. So, when you discuss performance and reliability you have to keep that in mind. In the past 15 years, the company has created new benchmarks

based on its services' and products' performance. The app engine provides the same reliability and performance as any other Google product.

Cost Savings

You don't have to hire engineers to manage your servers or to do that yourself. You can invest the money saved into other parts of your business.

Platform Independence

You can move all your data to another environment without any difficulty as there are not many dependencies on the app engine platform.

Programming Environment for Google

Creating a Google Cloud Platform project

To use Google's tools for your own site or app, you need to create a new project on Google Cloud Platform. This requires having a Google account.

- Go to the App Engine dashboard on the Google Cloud Platform Console and press the Create button.
- If you've not created a project before, you'll need to select whether you want to receive email updates or not, agree to the Terms of Service, and then you should be able to continue.
- Enter a name for the project, edit your project ID and note it down. For this tutorial, the following values are used:

  Project Name: GAE Sample Site

  Project ID: gaesamplesite

- **Click the Create button to create your project.**

Creating an application

- Each Cloud Platform project can contain one App Engine application. Let's prepare an app for our project.

- We'll need a sample application to publish. If you've not got one to use, download and unzip this sample app.
- Have a look at the sample application's structure — the website folder contains your website content and app.yaml is your application configuration file.

1) Your website content must go inside the website folder, and its landing page must be called index.html, but apart from that it can take whatever form you like.
2) The app.yaml file is a configuration file that tells App Engine how to map URLs to your static files. You don't need to edit it.

Publishing your application

Now that we've got our project made and sample app files collected together, let's publish our app.

1. Open Google Cloud Shell.
2. Drag and drop the sample-app folder into the left pane of the code editor.
3. Run the following in the command line to select your project:
    i. **gcloud config set project gaesamplesite**
4. Then run the following command to go to your app's directory:
    i. **cd sample-app**
5. You are now ready to deploy your application, i.e. upload your app to App Engine:
    i. **gcloud app deploy**
6. Enter a number to choose the region where you want your application located.
    i. **Enter Y to confirm.**
7. Now navigate your browser to your-project-id.appspot.com to see your website online. For example, for the project ID gaesamplesite, go to gaesamplesite.appspot.com.

App Engine

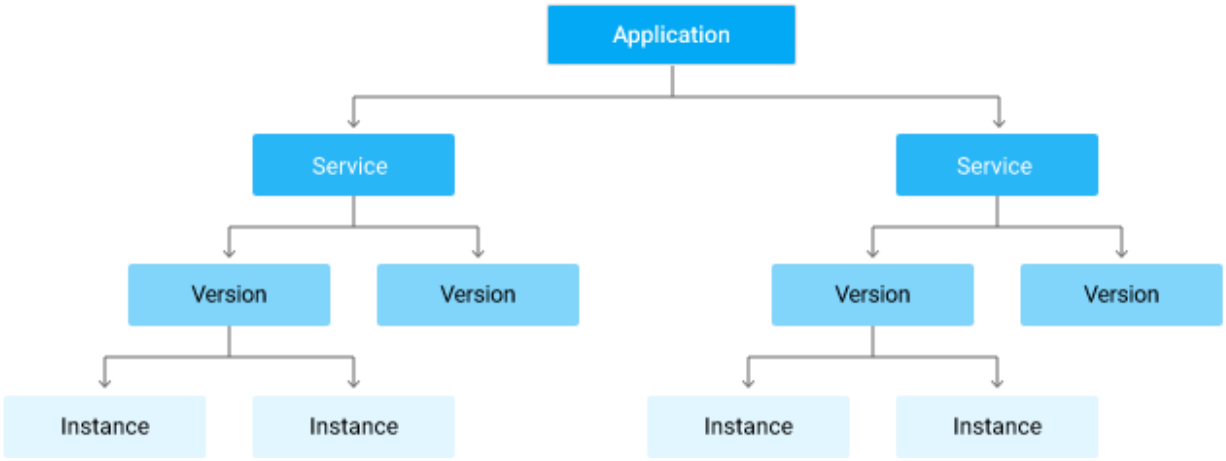
An App Engine app is made up of a single application resource that consists of one or more services. Each service can be configured to use different runtimes and to operate with different performance settings. Within each service, you deploy versions of that service. Each version then runs within one or more instances, depending on how much traffic you configured it to handle.

Components of an application

Your App Engine app is created under your Google Cloud project when you create an application resource. The App Engine application is a top-level container that includes the service, version, and instance resources that make up your app. When you create your App Engine app, all your resources are created in the region that you choose, including your app code along with a collection of settings, credentials, and your app's metadata.

Each App Engine application includes at least one service, the default service, which can hold as many versions of that service as you like.

The following diagram illustrates the hierarchy of an App Engine app running with multiple services. In this diagram, the app has two services that contain multiple versions, and two of those versions are actively running on multiple instances:



Services

Use services in App Engine to factor your large apps into logical components that can securely share App Engine features and communicate with one another. Generally, your App Engine services behave like microservices. Therefore, you can run your whole app in a single service or you can design and deploy multiple services to run as a set of microservices.

For example, an app that handles your customer requests might include separate services that each handle different tasks, such as:

- API requests from mobile devices
- Internal, administration-type requests
- Backend processing such as billing pipelines and data analysis

Each service in App Engine consists of the source code from your app and the corresponding App Engine configuration files. The set of files that you deploy to a service represent a single version of that service and each time that you deploy to that service, you are creating additional versions within that same service.

Versions

Having multiple versions of your app within each service allows you to quickly switch between different versions of that app for rollbacks, testing, or other temporary events. You can route traffic to one or more specific versions of your app by migrating or splitting traffic.

Instances

The versions within your services run on one or more instances. By default, App Engine scales your app to match the load. Your apps will scale up the number of instances that are running to provide consistent performance, or scale down to minimize idle instances and reduces costs. For more information about instances, see How Instances are Managed.

Application requests

Each of your app's services and each of the versions within those services must have a unique name. You can then use those unique names to target and route traffic to specific resources using URLs, for example:

https://VERSION_ID-dot-SERVICE_ID-dot-PROJECT_ID.REGION_ID.r.appspot.com

Incoming user requests are routed to the services or versions that are configured to handle traffic. You can also target and route requests to specific services and versions. For more information, see Handling Requests.

Logging application requests

When your application handles a request, it can also write its own logging messages to stdout and stderr. For details about your app's logs, see Writing Application Logs.

Limits

The maximum number of services and versions that you can deploy depends on your app's pricing:

Limit	Free app	Paid app
• Maximum services per app	5	105
• Maximum versions per app	15	210

Open Stack

OpenStack is a free open standard cloud computing platform, mostly deployed as infrastructure-as-a-service in both public and private clouds where virtual servers and other resources are made available to users.

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds. Backed by some of the biggest companies in software development and hosting, as well as thousands of individual community members, many think that OpenStack is the future of cloud computing. OpenStack is managed by the OpenStack Foundation, a non-profit that oversees both development and community-building around the project.

Introduction to OpenStack

OpenStack lets users deploy virtual machines and other instances that handle different tasks for managing a cloud environment on the fly. It makes horizontal scaling easy, which means that tasks that benefit from running concurrently can easily serve more or fewer users on the fly by just spinning up more instances. For example, a mobile application that needs to communicate with a remote server might be able to divide the work of communicating with each user across many different instances, all communicating with one another but scaling quickly and easily as the application gains more users.

And most importantly, OpenStack is open source software, which means that anyone who chooses to can access the source code, make any changes or modifications they need, and freely share these changes back out to the community at large. It also means that OpenStack has the benefit of thousands of

developers all over the world working in tandem to develop the strongest, most robust, and most secure product that they can.

How is OpenStack used in a cloud environment?

The cloud is all about providing computing for end users in a remote environment, where the actual software runs as a service on reliable and scalable servers rather than on each end-user's computer. Cloud computing can refer to a lot of different things, but typically the industry talks about running different items "as a service"—software, platforms, and infrastructure. OpenStack falls into the latter category and is considered Infrastructure as a Service (IaaS). Providing infrastructure means that OpenStack makes it easy for users to quickly add new instance, upon which other cloud components can run. Typically, the infrastructure then runs a "platform" upon which a developer can create software applications that are delivered to the end users.

What are the components of OpenStack?

OpenStack is made up of many different moving parts. Because of its open nature, anyone can add additional components to OpenStack to help it to meet their needs. But the OpenStack community has collaboratively identified nine key components that are a part of the "core" of OpenStack, which are distributed as a part of any OpenStack system and officially maintained by the OpenStack community.

Nova is the primary computing engine behind OpenStack. It is used for deploying and managing large numbers of virtual machines and other instances to handle computing tasks.

Swift is a storage system for objects and files. Rather than the traditional idea of a referring to files by their location on a disk drive, developers can instead refer to a unique identifier referring to the file or piece of information and let OpenStack decide where to store this information. This makes scaling easy, as developers don’t have the worry about the capacity on a single system behind the software. It also allows the system, rather than the developer, to worry about how best to make sure that data is backed up in case of the failure of a machine or network connection**.**

Cinder is a block storage component, which is more analogous to the traditional notion of a computer being able to access specific locations on a disk drive. This more traditional way of accessing files might be important in scenarios in which data access speed is the most important consideration.

Neutron provides the networking capability for OpenStack. It helps to ensure that each of the components of an OpenStack deployment can communicate with one another quickly and efficiently.

Horizon is the dashboard behind OpenStack. It is the only graphical interface to OpenStack, so for users wanting to give OpenStack a try, this may be the first component they actually "see." Developers can access all of the components of OpenStack individually through an application programming interface (API), but the dashboard provides system administrators a look at what is going on in the cloud, and to manage it as needed.

Keystone provides identity services for OpenStack. It is essentially a central list of all of the users of the OpenStack cloud, mapped against all of the services provided by the cloud, which they have permission to use. It provides multiple means of access, meaning developers can easily map their existing user access methods against Keystone.

Glance provides image services to OpenStack. In this case, "images" refers to images (or virtual copies) of hard disks. Glance allows these images to be used as templates when deploying new virtual machine instances.

Ceilometer provides telemetry services, which allow the cloud to provide billing services to individual users of the cloud. It also keeps a verifiable count of each user's system usage of each of the various components of an OpenStack cloud. Think metering and usage reporting.

Heat is the orchestration component of OpenStack, which allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application. In this way, it helps to manage the infrastructure needed for a cloud service to run.

Prerequisite for minimum production deployment

There are some basic requirements you'll have to meet to deploy OpenStack. Here are the prerequisites, drawn from the OpenStack manual.

Hardware: For OpenStack controller node, 12 GB RAM are needed as well as a disk space of 30 GB to run OpenStack services. Two SATA disks of 2 TB will be necessary to store volumes used by instances. Communication with compute nodes requires a network interface card (NIC) of 1 Gbps. For compute nodes, 2 GB RAM will be sufficient to run three tiny instances on a single compute node. Two NIC 1 Gbps will allow communication with both the controller and other compute nodes.

Operating system (OS): OpenStack supports the following operating systems: CentOS, Debian, Fedora, Red Hat Enterprise Linux (RHEL), openSUSE, SLES Linux Enterprise Server and Ubuntu. Other system support is provided by different editors or can be developed by porting nova modules on the target platform.

Federation in the Cloud

"Cloud federation manages consistency and access controls when two or more independent geographically distinct Clouds share either authentication, files, computing resources, command and control or access to storage resources."

Cloud federation introduces additional issues that have to be addressed in order to provide a secure environment in which to move applications and services among a collection of federated providers. Baseline security needs to be guaranteed across all cloud vendors that are part of the federation.

An interesting aspect is represented by the management of the digital identity across diverse organizations, security domains, and application platforms. In particular, the term federated identity management refers to standards-based approaches for handling authentication, single sign-on (SSO), role-based access control, and session management in a federated environment . This enables users to utilize services more effectively in a federated context by providing their authentication details only once to log into a network composed of several entities involved in a transaction. This capability is realized by either relying on open industry standards or openly published specifications (Liberty Alliance Identity Federation, OASIS Security Assertion Markup Language, and WS-Federation) such that interoperation can be achieved. No matter the specific protocol and framework, two main approaches can be considered:

Centralized federation model

This is the approach taken by several identity federation standards. It distinguishes two operational roles in an SSO transaction: the identity provider and the service provider.

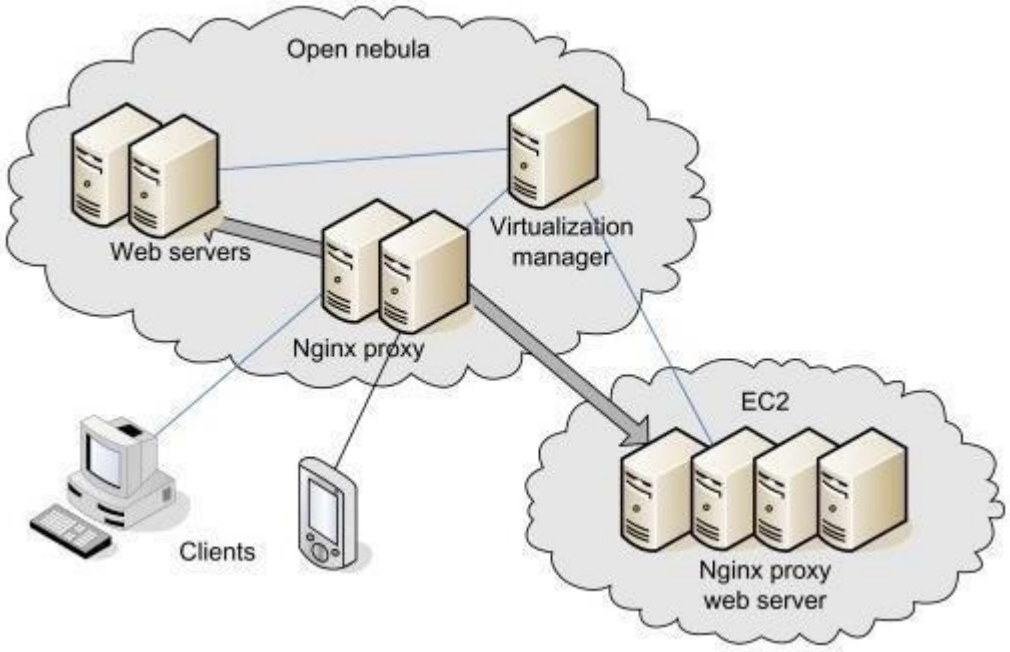
Claim-based model

This approach addresses the problem of user authentication from a different perspective and requires users to provide claims answering who they are and what they can do in order to access content or complete a transaction.

The first model is currently used today; the second constitutes a future vision for identity management in the cloud.

Digital identity management constitutes a fundamental aspect of security management in a cloud federation. To transparently perform operations across different administrative domains, it is of mandatory importance to have a robust framework for authentication and authorization, and federated identity management addresses this issue. Our previous considerations of security contribute to design and implement a secure system comprising the cloud vendor stack and the user application; federated identity management allows us to tie together the computing stacks of different vendors and present them as a single environment to users from a security point of view.

OpenNebula can be used in conjunction with a reverse proxy to form a cloud bursting hybrid cloud architecture with load balancing and virtualization support provided by OpenNebula . The OpenNebula VM controls server allocation in both the EC2 cloud as well as the OpenNebula cloud, while the Nginx proxy to which the clients are connected distributes load over the web servers both in EC2 as well as the OpenNebula cloud. In addition to web servers, the EC2 cloud also has its own Nginx load balancer.



Much research work has been developed around OpenNebula. For example, the University of Chicago has come up with an advance reservation system called Haizea Lease Manager. IBM Haifa has developed a policy-driven probabilistic admission control and dynamic placement optimization for site level management policies called the RESERVOIR Policy Engine, Nephele is an SLA-driven automatic service management tool developed by Telefonica and Virtual Cluster Tool for atomic cluster management with versioning with multiple transport protocols from CRS4 Distributed Computing Group.

Cloud Federations and Server Coalitions

In large-scale systems, coalition formation supports more effective use of resources, as well as convenient means to access these resources. It is therefore not surprising that coalition formation for computational grids has been investigated in the past. There is also little surprise that the interest in coalition formation migrated in recent years from computational grids to CRM. The interest in grid computing is fading away, while cloud computing is widely accepted today and its adoption by more and more institutions and individuals seems to be guaranteed at least for the foreseeable future.

Two classes of applications of cloud coalitions are reported in the literature:

1. Coalitions among CSPs for the formation of cloud federations. A cloud federation is an infrastructure allowing a group of CSPs to share resources; the goal is to balance the load and improve system reliability.
2. Coalitions among the servers of a data center. The goal is to assemble a pool of resources larger than the ones available from a single server.

In recent years the number of CSPs has increased significantly. The question if they should cooperate to share their resources led to the idea of cloud federations, groups of CSPs who have agreed on a set of common standards and are able to share their resources. The infrastructure of individual CSPs consists of a hierarchy of networks and millions of servers thus, a cloud federation would indeed be a very complex system.

The vast majority of ongoing research in this area is focused on game-theoretic aspects of coalition formation for cloud federations, while coalitions among the servers of a single cloud has received little attention in the past. This is likely to change due to the emerging interest in Big Data cloud applications which require more resources than a single server can provide. To address this problem, sets of identically configured servers able to communicate effectively among themselves form coalitions with sufficient resources for data- and computationally intensive problems.

Cloud coalition formation raises a number of technical, as well as nontechnical problems. Cloud federations require a set of standards. The cloud computing landscape is still evolving and an early standardization may slowdown and negatively affects the adoption of new ideas and technologies. At the same time, CSPs want to maintain their competitive advantages by closely guarding the details of their internal algorithms and protocols.

Reaching agreements on a set of standards is particularly difficult when the infrastructure of the members of the group is designed to support different cloud delivery models. For example, it is hard to see how the IaaS could be supported by either SaaS or PaaS clouds. Thus, in spite of the efforts coordinated by the National Institute of Standards (NIST), the adoption of inter-operability standards supporting cloud federations seems a rather distant possibility, that resource management in one cloud is extremely challenging therefore, dynamic resource sharing among multiple cloud infrastructures seems infeasible at this time. Communication between the members of a cloud federation would also require dedicated networks with low latency and high bandwidth.

Four Levels of Federation

Creating a cloud federation involves research and development at different levels: conceptual, logical and operational, and infrastructural.

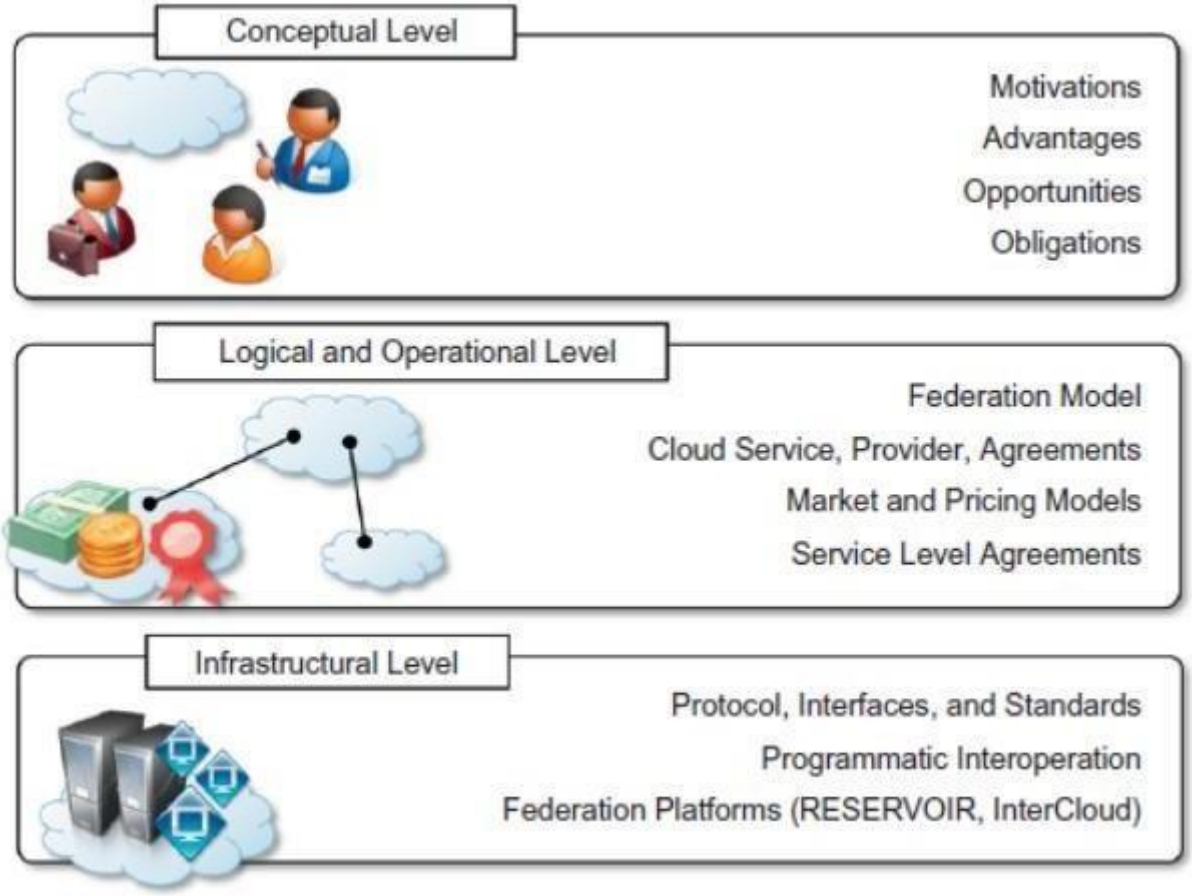


Figure provides a comprehensive view of the challenges faced in designing and implementing an organizational structure that coordinates together cloud services that belong to different administrative domains and makes them operate within a context of a single unified service middleware.

Each cloud federation level presents different challenges and operates at a different layer of the IT stack. It then requires the use of different approaches and technologies. Taken together, the solutions to the challenges faced at each of these levels constitute a reference model for a cloud federation.

CONCEPTUAL LEVEL

The conceptual level addresses the challenges in presenting a cloud federation as a favourable solution with respect to the use of services leased by single cloud providers. In this level it is important to clearly identify the advantages for either service providers or service consumers in joining a federation and to delineate the new opportunities that a federated environment creates with respect to the single-provider solution.

Elements of concern at this level are:

- Motivations for cloud providers to join a federation.
- Motivations for service consumers to leverage a federation.
- Advantages for providers in leasing their services to other providers.
- Obligations of providers once they have joined the federation.

- Trust agreements between providers.
- Transparency versus consumers.

Among these aspects, the most relevant are the motivations of both service providers and consumers in joining a federation.

LOGICAL & OPERATIONAL LEVEL

- The logical and operational level of a federated cloud identifies and addresses the challenges in devising a framework that enables the aggregation of providers that belong to different administrative domains within a context of a single overlay infrastructure, which is the cloud federation.

- At this level, policies and rules for interoperation are defined. Moreover, this is the layer at which decisions are made as to how and when to lease a service to—or to leverage a service from—another provider.
- The logical component defines a context in which agreements among providers are settled and services are negotiated, whereas the operational component characterizes and shapes the dynamic behaviour of the federation as a result of the single providers' choices.

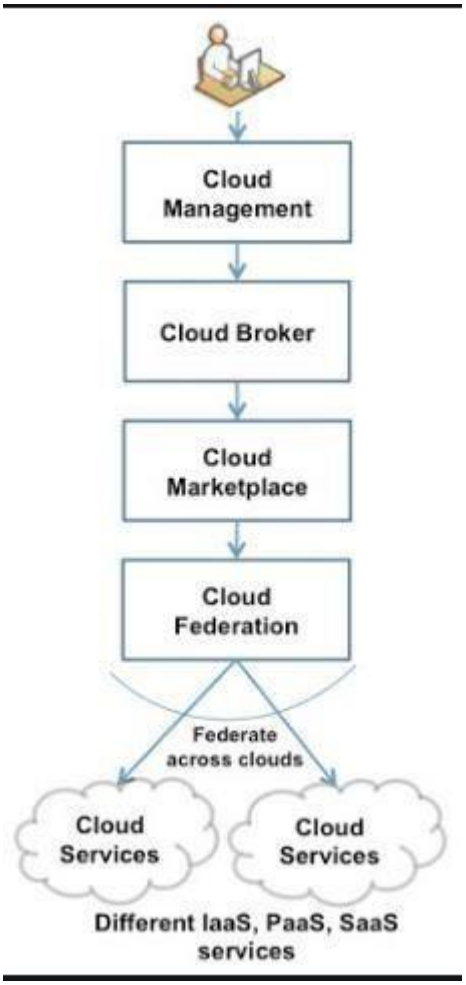
This is the level where MOCC is implemented and realized. It is important at this level to address the following challenges:

• How should a federation be represented?
• How should we model and represent a cloud service, a cloud provider, or an agreement?
• How should we define the rules and policies that allow providers to join a federation?
• What are the mechanisms in place for settling agreements among providers?
• What are provider's responsibilities with respect to each other?
• When should providers and consumers take advantage of the federation?
• Which kinds of services are more likely to be leased or bought?
• How should we price resources that are leased, and which fraction of resources should we lease? **The logical and operational level provides opportunities for both academia and industry.**

INFRASTRUCTURE LEVEL

The infrastructural level addresses the technical challenges involved in enabling heterogeneous cloud computing systems to interoperate seamlessly.

It deals with the technology barriers that keep separate cloud computing systems belonging to different administrative domains. By having standardized protocols and interfaces, these barriers can be overcome.



At this level it is important to address the following issues:

• What kind of standards should be used?

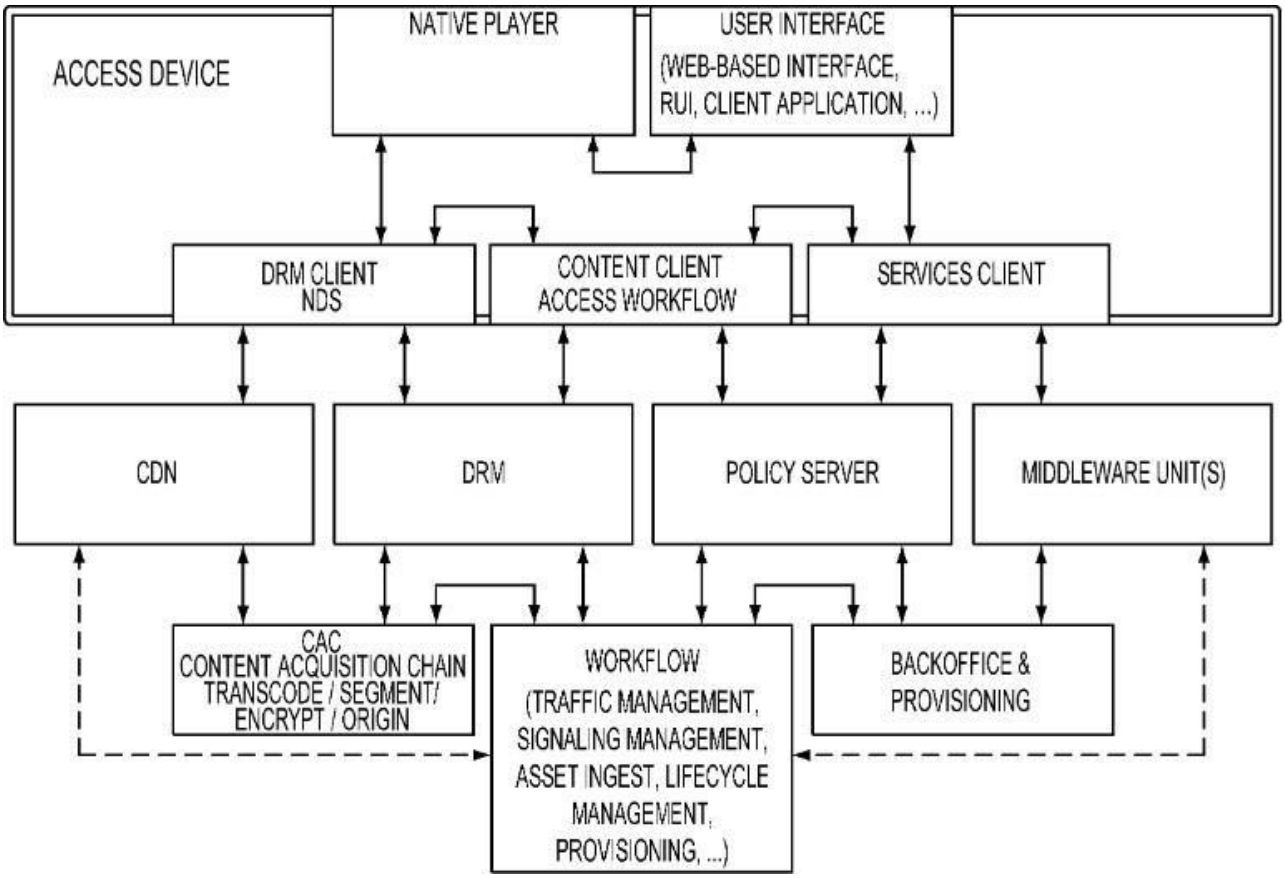
• How should design interfaces and protocols be designed for interoperation?

• Which are the technologies to use for interoperation?

• How can we realize a software system, design platform components, and services enabling interoperability?

Interoperation and composition among different cloud computing vendors is possible only by means of open standards and interfaces. Moreover, interfaces and protocols change considerably at each layer of the Cloud Computing Reference Model.

Federated Services and Applications



Future of Federation.

The federated cloud model is a force for real democratization in the cloud market. It's how businesses will be able to use local cloud providers to connect with customers, partners and employees anywhere in the world. It's how end users will finally get to realize the promise of the cloud. And, it's how data center operators and other service providers will finally be able to compete with, and beat, today's so-called global cloud providers.

The future of cloud computing as one big public cloud. Others believe that enterprises will ultimately build a single large cloud to host all their corporate services. This is, of course, because the benefit of cloud computing is dependent on large – very large – scale infrastructure, which provides administrators and service administrators and consumers the ability for ease of deployment, self service, elasticity, resource pooling and economies of scale. However, as cloud continues to evolve – so do the services being offered.

Cloud Services & Hybrid Clouds

Services are now able to reach a wider range of consumers, partners, competitors and public audiences. It is also clear that storage, compute power, streaming, analytics and other advanced services are best served when they are in an environment tailored for the proficiency of that service.

One method of addressing the need of these service environments is through the advent of hybrid clouds. Hybrid clouds, by definition, are composed of multiple distinct cloud infrastructures connected in a manner that enables services and data access across the combined infrastructure. The intent is to leverage the additional benefits that hybrid cloud offers without disrupting the traditional cloud benefits. While hybrid cloud benefits come through the ability to distribute the work stream, the goal is to continue to realize the ability for managing peaks in demand, to quickly make services available and capitalize on new business opportunities**.**

The Solution: Federation

Federation creates a hybrid cloud environment with an increased focus on maintaining the integrity of corporate policies and data integrity. Think of federation as a pool of clouds connected through a channel of gateways; gateways which can be used to optimize a cloud for a service or set of specific services. Such gateways can be used to segment service audiences or to limit access to specific data sets. In essence, federation has the ability for enterprises to service their audiences with economy of scale without exposing critical applications or vital data through weak policies or vulnerabilities.

- Many would raise the question: if Federation creates multiples of clouds, doesn't that mean cloud benefits are diminished? I believe the answer is no, due to the fact that a fundamental change has transformed enterprises through the original adoption of cloud computing, namely the creation of a flexible environment able to adapt rapidly to changing needs based on policy and automation.
- Cloud end-users are often tied to a unique cloud provider, because of the different APIs, image formats, and access methods exposed by different providers that make very difficult for an average user to move its applications from one cloud to another, so leading to a vendor lock-in problem.

- Many SMEs have their own on-premise private cloud infrastructures to support the internal computing necessities and workloads. These infrastructures are often over-sized to satisfy peak demand periods, and avoid performance slow-down. Hybrid cloud (or cloud bursting) model is a solution to reduce the on-premise infrastructure size, so that it can be dimensioned for an average load, and it is complemented with external resources from a public cloud provider to satisfy peak demands.

- Many big companies (e.g. banks, hosting companies, etc.) and also many large institutions maintain several distributed data-centers or server-farms, for example to serve to multiple geographically distributed offices, to implement HA, or to guarantee server proximity to the end user. Resources and networks in these distributed data-centers are usually configured as non-cooperative separate elements, so that usually every single service or workload is deployed in a unique site or replicated in multiple sites.

- Many educational and research centers often deploy their own computing infrastructures, that usually do not cooperate with other institutions, except in same punctual situations (e.g. in joint projects or initiatives). Many times, even different departments within the same institution maintain their own non-cooperative infrastructures.cloud federationThis Study Group will evaluate the main challenges to enable the provision of federated cloud infrastructures, with special emphasis on inter-cloud networking and security issues:

- Security and Privacy
- Interoperability and Portability
- Performance and Networking Cost

It is important to bring perspectives from Europe and USA in order to define the basis for an open cloud market, addressing barriers to adoption and meeting regulatory, legal, geographic, trust and performance constraints.

This group will directly contribute to the first two key actions of the European Cloud Strategy ”Unleashing the Potential of Cloud Computing in Europe”.

The **first key action** aims at “Cutting through the Jungle of Standards” to help the adoption of cloud computing by encouraging compliance of cloud services with respect to standards and thus providing evidence of compliance to legal and audit obligations. These standards aim to avoid customer lock in by promoting interoperability, data portability and reversibility.

The **second key action** “Safe and Fair Contract Terms and Conditions” aims to protect the cloud consumer from insufficiently specific and balanced contracts with cloud providers that do not “provide for liability for data integrity, confidentiality or service continuity”. The cloud consumer is often presented with "take-it-or-leave-it standard contracts that might be cost-saving for the provider but is often undesirable for the user”. The commission aims to develop with “stakeholders model terms for cloud computing service level agreements for contracts”.

Server to Server Sharing

The first version of this ideas was implemented in ownCloud 7.0 as “Server to Server Sharing”. ownCloud already knew the concept of sharing anonymous links with people outside of the server. And, as ownCloud offered both a WebDAV interface and could mount external WebDAV shares, it was possible to manually hook a ownCloud into another ownCloud server. Therefore the first obvious step

was to add a “Add to your ownCloud” button to this link shares, allowing people to connect such public links with their cloud by mounting it as a external WebDAV resource.

Interface: Various cloud service providers have different APIs, pricing models and cloud infrastructure. Open cloud computing interface is necessary to be initiated to provide a common application programming interface for multiple cloud environments. The simplest solution is to use a software component that allows the federated system to connect with a given cloud environment. Another solution can be to perform the federation at Infrastructure level and not at application or service level.

Networking: Virtual machines in the cloud may be located in different network architectures using different addressing schemes. To interconnect these VMs a virtual network can be formed on the underlying physical network with uniform IP addressing scheme. When services are running on remote clouds, main concern is security of the sensitive strategic information running on remote cloud. C. Heterogeneity of resource: Each cloud service providers offers different VMs with varying processing memory and storage capacity resulting in unbalanced processing load and system instability. It is likely that the cloud owner will purchase latest models of hardware available at the time of purchase while it is unlikely to retire the older model nodes until their useful life is over. This creates heterogeneity.

Federated Cloud Sharing

Server to server sharing already helped a lot to establish some bridges between many small islands created by the ability to self-host your cloud solution. But it was still not the kind of integration people where used to from the large centralized services and it only worked for ownCloud, not across various open source file sync and share solutions.

Trusted Servers

In order to make it easier to find people on other servers we introduced the concept of “trusted servers” as one of our last steps. This allows administrator to define other servers they trust. If two servers trust each other they will sync their user lists. This way the share dialogue can auto-complete not only local users but also users on other trusted servers. The administrator can decide to define the lists of trusted servers manually or allow the server to auto add every other server to which at least one federated share was successfully created. This way it is possible to let your cloud server learn about more and more other servers over time, connect with them and increase the network of trusted servers.

Open Challenges: where we’re taking Federated Cloud Sharing

Of course there are still many areas to improve. For example the way you can discover users on different server to share with them, for which we’re working on a global, shared address book solution. Another

point is that at the moment this is limited to sharing files. A logical next step would be to extend this to many other areas like address books, calendars and to real-time text, voice and video communication and we are, of course, planning for that.