1. **With the help of a neat diagram explain the architecture of virtualized data center.**

Ans.:

Refer Que. No. 21

2. **Explain public, private and hybrid clouds.**

Ans.:

- The following figure shows, both public clouds and private clouds are developed in the Internet.

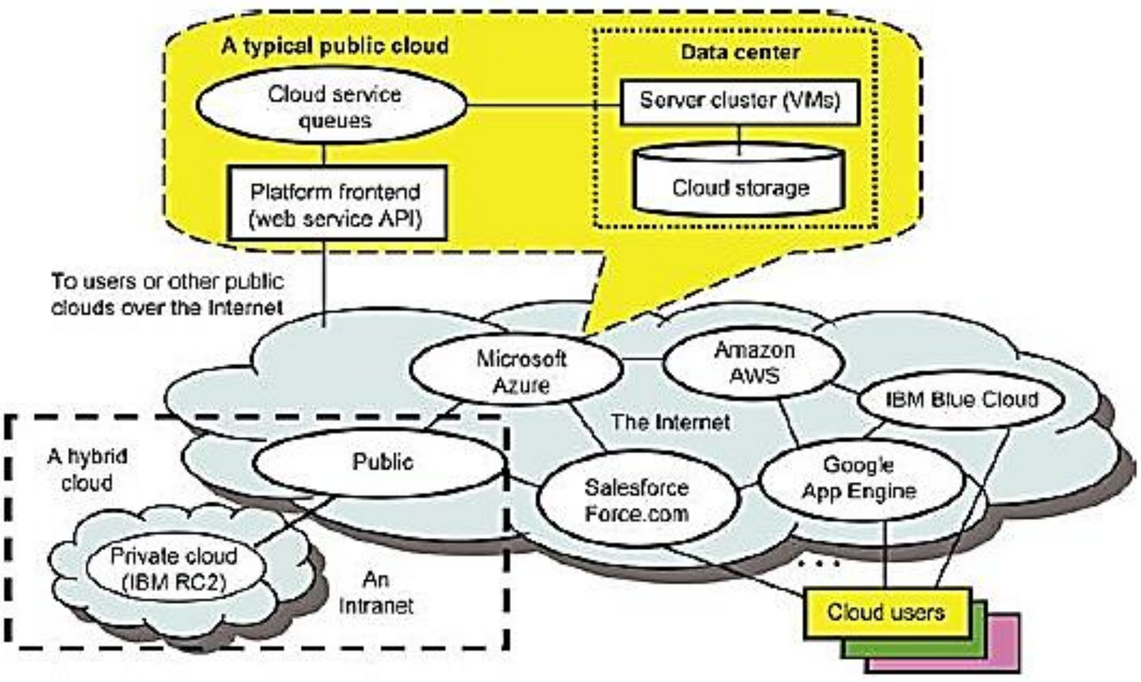


Fig: Public, private, and hybrid clouds illustrated by functional architecture and connectivity of representative clouds available by 2011.

- ❖ **Public Clouds:**

- ➢ A public cloud is built over the Internet and can be accessed by any user who has paid for the service.
- ➢ Public clouds are owned by service providers and are accessible through a subscription. The callout box in top of Figure shows the architecture of a typical public cloud.
- ➢ Many public clouds are available, including Google App Engine (GAE), Amazon Web Services (AWS), Microsoft Azure, IBM Blue Cloud, and Salesforce.com's Force.com.
- ➢ The providers of the aforementioned clouds are commercial providers that offer a publicly accessible remote interface for creating and managing VM instances within their proprietary infrastructure.

- A public cloud delivers a selected set of business processes. The application and infrastructure services are offered on a flexible price-per-use basis.

- **Private Clouds:**
- A private cloud is built within the domain of an intranet owned by a single organization.
- Therefore, it is client owned and managed, and its access is limited to the owning clients and their partners.
- Private clouds give local users a flexible and agile private infrastructure to run service workloads within their administrative domains.
- A private cloud is supposed to deliver more efficient and convenient cloud services.
- It may impact the cloud standardization, while retaining greater customization and organizational control.

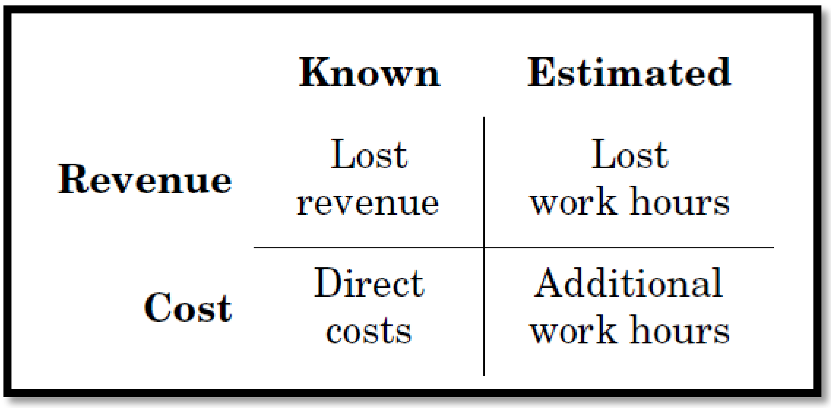
- **Hybrid Clouds:**
- A hybrid cloud is built with both public and private clouds, as shown at the lower-left corner of Figure.
- Private clouds can also support a hybrid cloud model by supplementing local infrastructure with computing capacity from an external public cloud.
- For example, the Research Compute Cloud (RC2) is a private cloud, built by IBM, that interconnects the computing and IT resources at eight IBM Research Centers scattered throughout the United States, Europe, and Asia.
- A hybrid cloud provides access to clients, the partner network, and third parties. In summary, public clouds promote standardization, preserve capital investment, and offer application flexibility.
- Private clouds attempt to achieve customization and offer higher efficiency, resiliency, security, and privacy.
- Hybrid clouds operate in the middle, with many compromises in terms of resource sharing.

3. **Discuss the business consequences of IT outages.**

Ans.:

- For a business, an IT outage is not the real issue, but the consequences that are associated with it are.
- As Fig. shows, IT outages affect either revenues or costs and we either can determine the effect directly or we can estimate it.

- **Direct costs** are associated with repair of IT defects, needed to continue IT operations.
- Devices need to be repaired, shipping has to be paid for, external consultants might be needed, etc.
- Other direct costs are contract penalties that have to be paid if an IT outage causes a delay in delivery of a service, beyond a contractual obligation.
- **Additional work hours** are indirect costs that are attributed to any incident.
- IT staff will put work into remedying any IT problems, and that work has to be paid for by the company, in the end, by sales.
- IT outages may result in additional work hours in other areas of company as well. The delivery staff might need additional work hours if the inventory system is down.
- The office workers will put up with evening work to get the work done that they could not do when they were not able to access any files, addresses, or emails during normal working hours.



	Known	**Estimated**
Revenue	Lost revenue	Lost work hours
Cost	Direct costs	Additional work hours

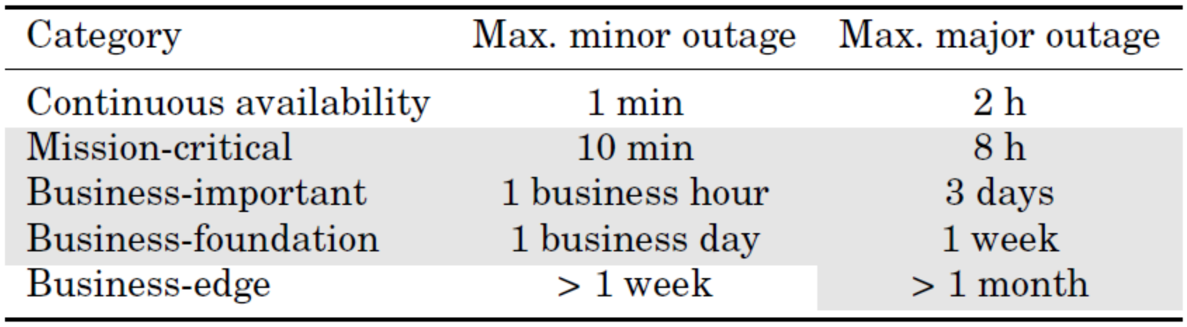
Fig: Business consequences of outages

- **Lost work hours** are indirect indicators for lost revenue. When 1000 office workers cannot work for 2 h because some server is down, or when goods cannot be delivered, the sales that could have been made in that time span might be lost forever.
- **Lost revenue** may also be directly attributed to an IT outage. This is the most important business consequence, but it is the hardest to measure directly.
- In the end, it is easier to specify costs than lost revenues as outage consequences. Costs can be identified and counted; lost revenues can only be estimated.

4. Explain the different categories of system and outage.

Ans.:

- The different categories of system and outages are described in following table:



Category	Max. minor outage	Max. major outage
Continuous availability	1 min	2 h
Mission-critical	10 min	8 h
Business-important	1 business hour	3 days
Business-foundation	1 business day	1 week
Business-edge	> 1 week	> 1 month

Table: System and outage categories

- There are a few fine points in the table that need further elaboration.
- ➢ First of all, the outage times are per incident. For minor outage requirements, we will need additional values. These additional values will spell out the allowed cumulated outage times over a certain time span, e.g., over a month or a year.
- ➢ It must be emphasized that these are maximum outage times. This means that they are limits for the respective category. For example, the entries in that table demand that major outages for mission critical systems must not last longer than 3 days. Everything above 10 min is considered a major outage.
- ➢ The time spans are not just absolute times, they take into account if something happens during or outside business hours. Very few of our IT systems need to run all the time; often we were able to specify business hours where availability requirements are much more stringent than for the wee hours.
- ➢ There are two ways to approach this table. One might have a vague notion that some systems are very important, more important, or less important for one's company. Analysis of the situation helps to associate outage limits with that intuitive categorization. On the other hand, one might have several SLAs already that name such limits, then the table helps to define good categories.

5. What is High availability? Explain in detail.

Ans.:

- High availability is the characteristic of a system to protect against or recover from minor outages in a short time frame with largely automated means.
- There are 3 factors when we talk about high availability:

1. **Outage categorization**: This is a precondition that tells us if we are in that problem and solution domain at all. We need to know potential failure scenarios for a service and the minor outage requirements for them. Only then can we start to talk about high availability.
2. **System categorization**: That tells us about requirements for maximum outage times. Only when those times are short do we speak of high availability. When a system can be down for a whole week, high availability is not involved.
3. **Automated protection or recovery**: Technology and solution approaches also have an influence if we need high availability. The same requirement may be resolved for two different services in two different ways: one needs high-availability technology, the other does not.

- High-availability design ensures that as few resources as possible are necessary as single component instances.
- We need to differentiate generic high availability from continuous availability which implies nonstop, uninterrupted service.
- Continuous availability is a subset of high availability where every component failure is protected against, and no after failure recovery takes place.

6. Discuss availability, reliability and serviceability of a system.

Ans.:

Availability:

- Availability is the measure of how often or how long a service or a system component is available for use.
- Availability also means features which help the system to stay operational even if failures occur. For example, mirroring of disks improves availability.
- The base availability measurement is the ratio of uptime to total elapsed time:

$$\text{availability} = \frac{\text{uptime}}{\text{uptime} + \text{downtime}} .$$

- The same availability can be expressed in absolute numbers (239 of 240 h last month) or as a percentage (99.6% last month).
- It can also be expressed in user-related terms where the time span is multiplied with the number of users.
- This is the actual availability of a system that can be measured for its existence.
- For identical systems, experience with old systems can be reused as a planning guide. Otherwise, if one knows the mean time between failures (MTBF) and the mean time to repair (MTTR), one can express planned or expected availability as

$$\text{availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} .$$

Reliability:

- Reliability is a measurement of fault avoidance. It is the probability that a system is still working at time *t*+1 when it worked at time *t*.
- A similar definition is the probability that a system will be available over a time interval *T*.
- Reliability does not measure planned or unplanned downtimes; MTTR values do not influence reliability. Reliability is often expressed as the MTBF.
- When we run thousands of disks, the MTBF for one disk becomes meaningful. But for computer systems, the MTBF has only a restricted value.
- Reliability features help to prevent and detect failures.

Serviceability:

- Serviceability is a measurement that expresses how easily a system is serviced or repaired. For example, a system with modular, hot-swappable components would have a good level of serviceability.
- It can be expressed as the inverse amount of maintenance time and number of crashes over the complete life span of a system.
- Like availability, there are two measurements that are of interest: planned and actual serviceability.
- Planned serviceability is a requirement that goes into the architecture as a design objective.
- For example, the planned serviceability of the same system might be 9-h planned service in 720-h elapsed time.

- Good serviceability is directly coupled to good patch and deployment management.
- Serviceability features help to identify failure causes, system diagnosis to detect problems before failures occur, simplify repair activities, and speed them up.
- A call-home feature and hot-swappable components are examples of serviceability features. Good serviceability increases both availability and reliability.

7. What is disaster recovery? Explain in detail.
Ans.:

- Traditionally, disaster recovery describes the process to survive catastrophes in the physical environment: fires, floods, hurricanes, earthquakes, terrorist attacks.
- Disaster recovery is the ability to continue with services in the case of major outages, often with reduced capabilities or performance. Disaster-recovery solutions typically involve manual activities.
- Disaster recovery handles the disaster when either a single point of failure is the defect or when many components are damaged and the whole system is rendered un-functional.
- It handles the case when operations cannot be resumed on the same system or at the same site.
- Instead, a replacement or backup system is activated and operations continue from there.
- This backup system may be on the same site as the primary system, but is usually located at another place.
- Therefore, for each IT system, we need to define the situation of a disaster.
- The classification of major outage and associated data loss is used to describe the objectives of disaster recovery:
  - ➢ **Recovery time objective (RTO):** The time needed until the service is usable again after a major outage.
  - ➢ **Recovery point objective (RPO):** The point in time from which data will be restored to be usable. In disaster cases, often some part of work is lost.
- Declaring a disaster and migrating to the backup system has grave consequences. This is an expensive and sometimes risky operation, and migrating back to the primary system will be expensive again.

- Data loss might happen, in the realm of our RPO. Therefore one has to be very cautious about disaster declaration and triggering disaster recovery.
- Disaster recovery is always associated with a bigger risk than high availability precautions. It is rarely utilized, and is usually tested infrequently.

8. Explain the spiral model of high availability and disaster recovery implementation.

Ans.:

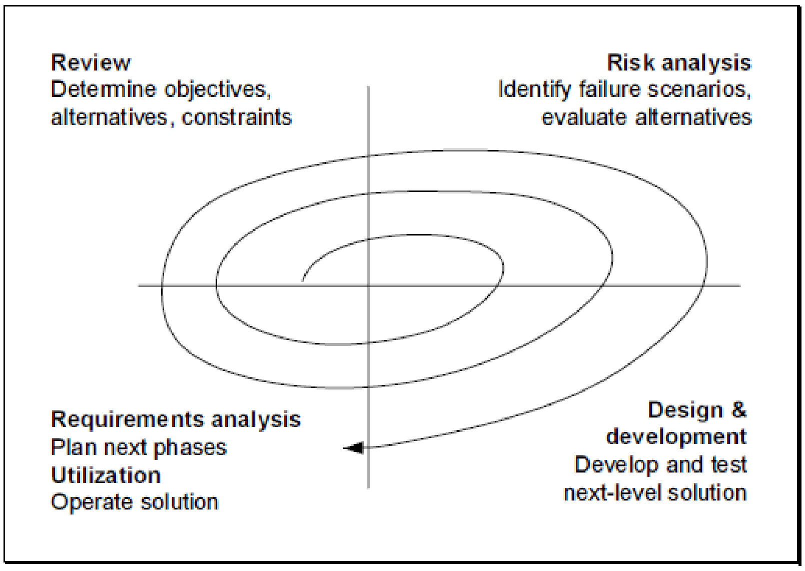

Review
Determine objectives, alternatives, constraints
Risk analysis
Identify failure scenarios, evaluate alternatives
Requirements analysis
Plan next phases
Utilization
Operate solution
Design & development
Develop and test next-level solution

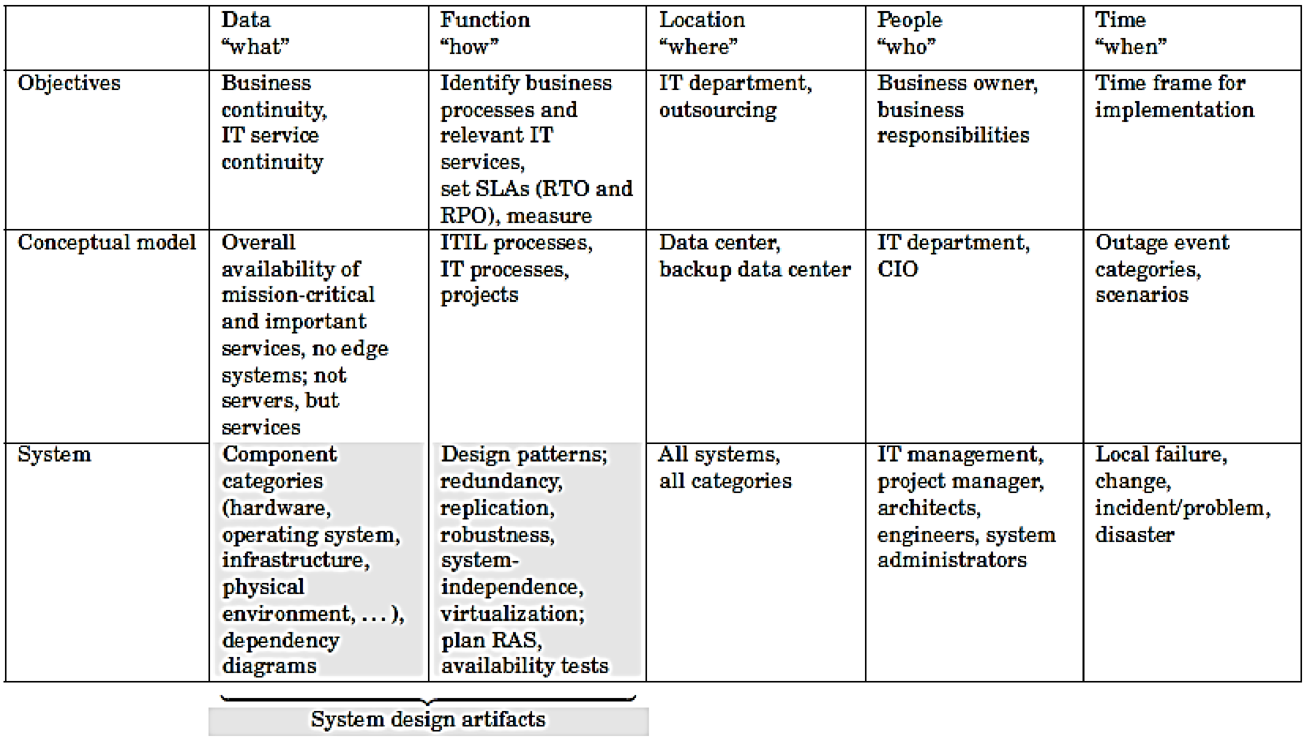
Fig: Spiral model of high-availability and disaster-recovery implementation

- The creation of highly available systems and to handle disasters we need to:
- ➢ Analyse our existing systems, their environment, and their usage
- ➢ Identify potential failure scenarios
- ➢ Design the future systems to continue operations in spite of these failures
- On a very high abstraction level, our approach has two tiers that are simple to explain:
- ➢ **Robustness**: We will minimize the potential for failures.
- ➢ **Redundancy**: We will establish resources to continue operations when failures occur.
- These approaches are not independent; in fact, they conflict often enough.
- Adding redundancy makes a system more complex and usually reduces its robustness.
- That is because the redundant component introduces new dependencies; it must be operated, it can fail as well, its failure can propagate to other components, etc.

- Therefore, we need to take great care to design our redundancy and use it only where it makes sense.
- As guidance, we utilize a scheme of component categories where failure may occur, from the physical environment to hardware to the user environment. For each component, potential failure causes exist that can be analysed in turn.
- Components do not exist in an isolated space. Most of them depend in their functionality on another component. On a higher abstraction level, we can express that as a dependency between the categories.
- One can express such component relationships by dependency diagrams. These diagrams are not arbitrary graphs, instead they show a hierarchy of categories, where higher layers depend on lower layers.
- The user environment depends on functional applications, they depend on middleware, which depend on infrastructure and operating systems, and so on.
- These categories form a kind of layered stack structure. For one, this is not a strictly hierarchical stack; applications use middleware, infrastructure, and operating systems, administrators work on and with several components too.
- Second, (network) infrastructure services like routers, Domain Name Service (DNS), or license servers.
- Such a classification is extremely valuable, and it must not be underestimated.
- We also need to select appropriate failure scenarios. As planning high availability and disaster recovery occurs in a business context. That business context also determines, or at least influences, the project’s scope.
- This selection will be influenced by many factors, both hard ones and soft ones.
- The “hard factors” are those that can be easily counted and measured, “soft factors” are experience and judgment calls that can be specified but where measurement becomes difficult.
- Soft factors are mostly human-related. Familiarity of our IT staff with certain solutions must certainly be considered.
- Setting up a high availability system for a mission-critical server is no place to try out something completely new.
- Other soft factors are vendor relationship – if we have a vendor that provides an excellent service, it makes good business sense to go with that vendor as long as it does not result in a vendor lock-in situation.

- All those factors will have to be revisited and re-evaluated regularly.
- Changing requirements, changing technology, and new possibilities all influence the decision process. Sometime down the road, a formerly well-reasoned decision might need reworking.
- This leads to a spiral model of requirements analysis, design, development, utilization, and review which is illustrated in above figure.

9. Explain the high-availability and disaster recovery architecture.
Ans.:



	Data "what"	Function "how"	Location "where"	People "who"	Time "when"
Objectives	Business continuity, IT service continuity	Identify business processes and relevant IT services, set SLAs (RTO and RPO), measure	IT department, outsourcing	Business owner, business responsibilities	Time frame for implementation
Conceptual model	Overall availability of mission-critical and important services, no edge systems; not servers, but services	ITIL processes, IT processes, projects	Data center, backup data center	IT department, CIO	Outage event categories, scenarios
System	Component categories (hardware, operating system, infrastructure, physical environment, . . .), dependency diagrams	Design patterns; redundancy, replication, robustness, system-independence, virtualization; plan RAS, availability tests	All systems, all categories	IT management, project manager, architects, engineers, system administrators	Local failure, change, incident/problem, disaster

System design artifacts

Table: Overview of high-availability and disaster-recovery architecture. Abstraction levels are from top to bottom and aspects are from left to right

- An architecture is a two-dimensional endeavour where certain aspects are described for different scopes or different abstraction levels.
- The aspects are:
  - **Data**: What is the architecture concerned with, on the respective abstraction level?
  - **Function**: How is the data worked with, or how is a functionality to be achieved?
  - **Location**: Where is the data worked with, or where is the functionality achieved?

  - **People**: Who works with the data and achieves the functionality? Who is responsible, who approves, who supports?
  - **Time**: When is the data processed, or when is the functionality achieved?
- Each aspect can be described for each of the following abstraction levels:
  - **Objectives**: What shall this architecture achieve? How shall it be done, on an organizational level? Which organizations are responsible?
  - **Conceptual model**: Realization of the objectives on a business process level. Explanation of how the business entities work together in business locations on business processes, using work flows and their schedules.
  - **System model**: The logical data model and the application functions that must be implemented to realize the business concepts. The roles, deliverables, and processing structures to do so.

10. Explain the following terms with respect to disaster recovery:

i. Major outage or Disaster:

Major outage: A failure that impacts IT service for end users and which cannot be repaired within the availability limits of the SLA.
Major outages are also covered by SLAs, in the "Service Continuity" section, and are described with the recovery time objective (RTO) and the recovery point objective (RPO).

Disaster: a disaster is a synonym for a major outage. Both terms are used interchangeably.

ii. Declaration of disaster:

The decision that a major outage has happened and that disaster recovery starts.
This decision is usually not done automatically, but is made by IT staff and business owners together, as part of an escalation management process.
Often the affirmation of executive management is necessary to declare a disaster.

iii. Recovery time objective:

The time needed until the IT service is usable again after a major outage.
It starts with the declaration of disaster and is part of the SLA that describes the handling of major outages.

iv. Recovery point objective:

The point in time from which data will be restored to be usable.
It is typically expressed as a time span before the declaration of disaster and, like the RTO, is part of the SLA that describes the handling of major outages.
In major outages, often some part of work is lost.

v. Disaster recovery:
The process to restore full functionality in the case of major outages, including all necessary preparation actions.

vi. Disaster recovery planning:
The management activity to define the necessary actions for disaster recovery and that governs their implementation.

vii. Primary and disaster recovery systems:
In normal operation, the primary system supplies the IT service. A disaster-recovery system takes over functionality and supplies the service in the case of a major outage.

viii. Primary and disaster recovery sites:
The prototypical disaster scenario is destruction of the physical environment, e.g., by floods or fire. The site where the primary IT systems are normally placed is called the **primary site**.
A **disaster-recovery site** is a site where disaster recovery systems are placed. The disaster-recovery site is at a location that is (hopefully) not covered by the disaster and can take over the role of the primary site during disaster recovery.
Primary and disaster-recovery sites are roles, not absolute descriptions. A disaster-recovery site for one service may very well be the primary site for another service, and vice versa. In fact, this is a very common setup.

11. Discuss the principal approach to plan and realize disaster recovery.
Ans.:

- The principal approach to plan and realize disaster recovery was the same approach that is also used to plan and realize high availability.
- The approach is as follows:

1. Determine objectives
(a) Identify systems that need disaster recovery
(b) Determine RTO and RPO
(c) Identify users and departments that are affected
(d) Identify responsibilities of IT staff and business owners

2. Create conceptual design

(a) Create changes to business and IT processes that are necessary to realize and support disaster recovery
(b) Identify and set involved system locations, i.e., primary and disaster-recovery sites
(c) Define high-level solution approach and associated costs
(d) Determine IT staff and vendors that are involved; create RASIC chart to define roles and responsibilities
(e) Make a first stub at failure scenarios and failure categories, from a business point of view
(f) Evaluate scenarios, and determine their relative probability and relative damage
(g) Determine which scenarios are already covered by high-availability technology, which must be covered by this disaster-recovery project, and which are out of scope

3. Create system design

(a) Analyse the system components, create a project-specific system stack, and create a dependency diagram
(b) Extend failure scenarios to cover technical component failures
(c) Find single points of failure. Either provide recovery for them through redundancy, or provide other means to restore acceptable service within the disaster-recovery SLAs.
(d) Review that the solution handles all relevant scenarios

4. Implement solution

(a) Technology for implementation
(b) Technology that is particular to disaster recovery
(c) Particulars of network design for disaster recovery

5. Define process

(a) Create detailed disaster recovery procedures
(b) Train people
(c) Specify and conduct tests.

6. Update design and implementation as necessary

(a) When objectives change
(b) When primary systems are changed or updated
(c) When experience from other projects and problem analysis show potential improvements

12. Discuss the scope of disaster recovery.

Ans.:

- Disaster recovery focuses on essential business functionality and the minimum IT resources required for that.
- As disaster scenarios are not a common case, it will usually not be necessary to provide redundancy for full operations.
- Therefore some components might be declared out of scope for disaster recovery. Other components are in scope; they may be applications, computer systems, or whole sites.
- For disaster-recovery planning, the central questions are:
  - ➢ What needs to be protected (services/data)?
  - ➢ Which failures are connected with which disaster?
  - ➢ What essential functionality must stay operative with the highest priority?
  - ➢ What functionality must be made available again as soon as possible?
  - ➢ Which minimum IT resources are required for this?
- From a strategic point of view, the answers to these questions follow business demands.
- From a technical point of view, however, it is not enough to know which business services shall be protected or re-established.
- Therefore, it is necessary to know which applications, servers, and sites are relevant for which business service.

➢ ***Server Availability***

- To handle disaster recovery for servers, the following information should be known about a server:
  - ✓ Technical data: site location, architecture, capacity, maintenance contracts.
  - ✓ Which applications and infrastructure services are provided for which business service?
  - ✓ Which redundancy and failover solutions are available?
  - ✓ Which backup strategies are available?

➢ ***Application Availability***

- To handle disaster recovery for applications, the following information should be known about an application:
  - ✓ Which servers are involved for functionality of the application?
  - ✓ For which business service is this application essential?
  - ✓ Which other applications are needed for functionality?

➢ ***Site Availability***

- To handle disaster recovery for a site, the following information should be known:
  - ✓ Which servers are located on the site?
  - ✓ Which high-availability solutions (redundancy, failovers) are available?

- To control the disaster-recovery process, the information just listed should be available.
- An essential help is a dependency diagram of the primary system that shows the dependencies between sites, servers and applications.

13. Explain state synchronization with respect to disaster recovery.
Ans.:

- Disaster recovery is concerned with setup, maintenance, and operation of disaster-recovery systems at disaster-recovery sites.
- To establish IT functionality on the disaster recovery system, the system must have all applications, necessary configurations, and all data.
- Especially data from the primary system must be available at the disaster-recovery system in a short time frame, otherwise too much is lost.
- In the end, it boils down to our wanting to replicate all changes from the primary system on the disaster-recovery system, so that the disaster-recovery system is in the same state as the primary one.
- In other words, we want state synchronization for data, installation, and configuration.
- In practice, state synchronization happens either on the operating system level of volume managers and mirrored disk volumes or on higher levels: many middleware servers (databases or application servers) provide replication functionality that can be utilized.
- Sometimes it is also possible to replicate the configuration or data if it is stored in files.
- Many systems demand the properties atomicity, consistency, isolation, and durability (ACID). Enterprise-strength relational database management systems (RDBMS) are usually used to supply them.
- This implies that disaster-recovery architecture and implementation must support failover of RDBMS operations.

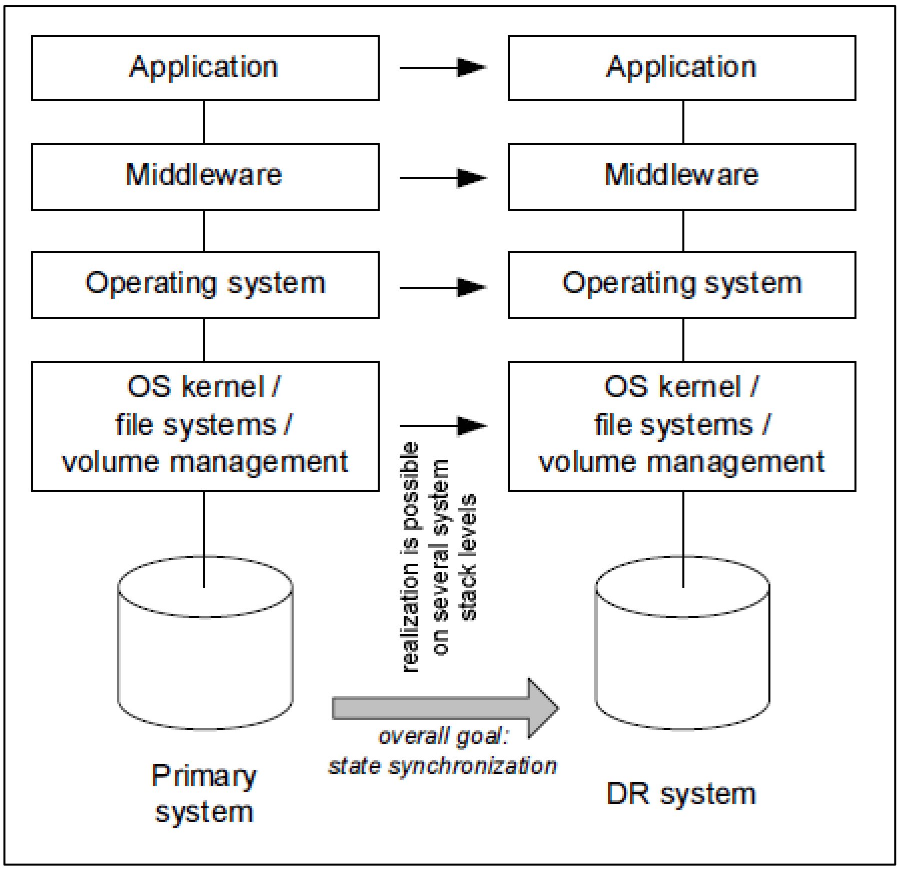

Application
Application
Middleware
Middleware
Operating system
Operating system
OS kernel / file systems / volume management
OS kernel / file systems / volume management
realization is possible on several system stack levels
overall goal: state synchronization
Primary system
DR system

Fig: Possible realization of state synchronization

- A method is needed to check the data for consistency and integrity, otherwise disk or application data mirroring might start the disaster-recovery system in an inconsistent and unrecoverable state, leading to software errors and crashes of the disaster recovery system.
- Care must also be taken to ensure that the available network bandwidth is sufficient to use the disaster-recovery system.
- Another important consideration is the requirement that the primary system must not be affected by the synchronization.

14. What are the three basic types of system designs for disaster recovery? Explain.

Ans.:

- There are three basic types of disaster-recovery system designs as follows:

1. **Shared systems**, where all available systems are used and the workload is distributed differently in the case of problems.
2. **Hot standby**, where an unused disaster-recovery system is ready to run all the time, and is kept on the same software, configuration, and data level as the primary system.

3. **Cold standby**, where an unused disaster-recovery system is available but is not up. In the case of a disaster, the disaster-recovery system is restored from a backup system to an appropriate state and started.

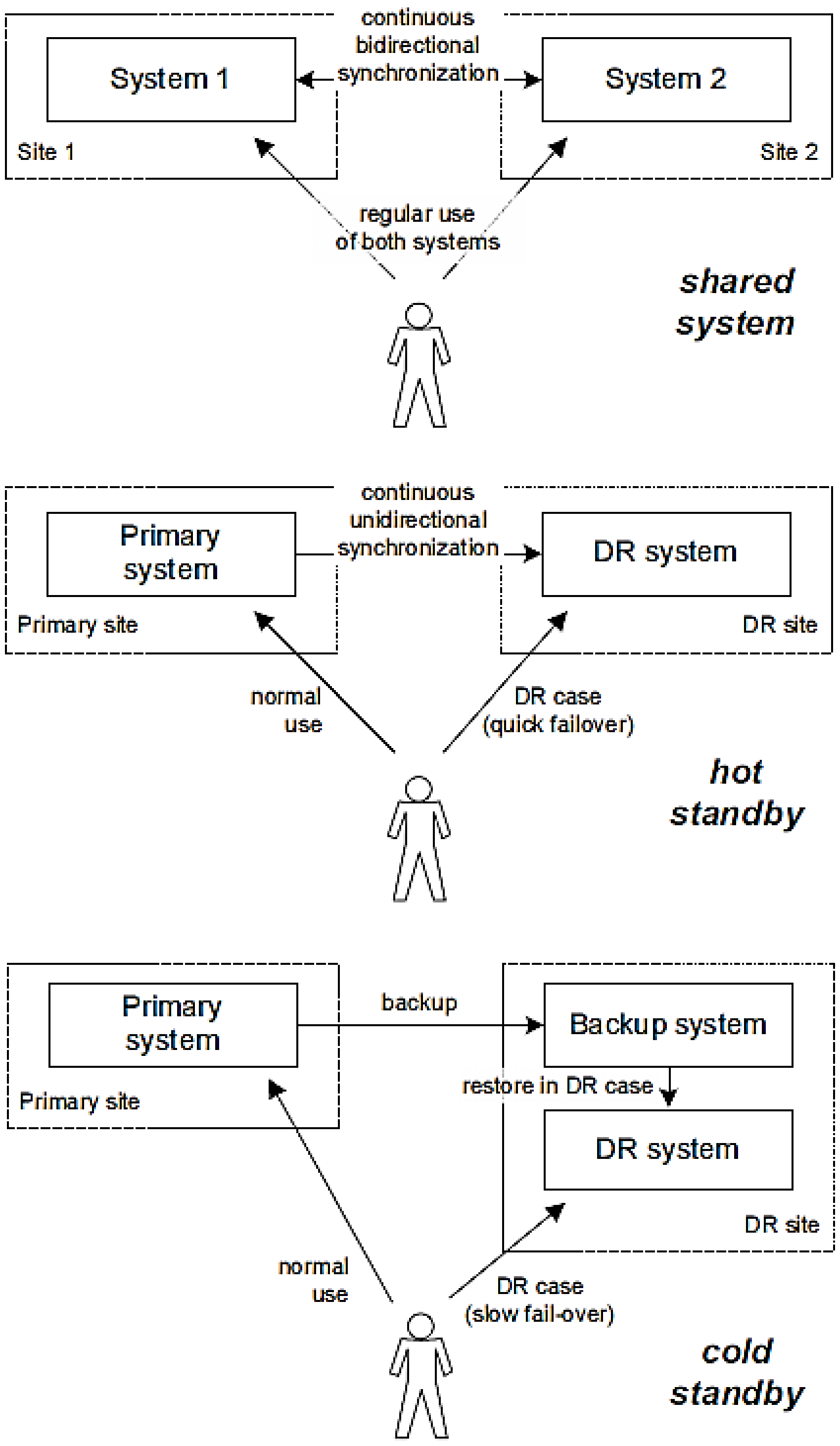


Fig: Disaster-recovery system designs

- There are two subcategories of cold-standby disaster-recovery systems.

  - The first one has systems where all the software needed is installed already and only the data is missing.
  - The second one uses completely unrelated systems, provisions them anew in the disaster-recovery case with all necessary software and configurations, and then restores data from a backup.
- Design types 2 and 3 use dedicated disaster-recovery systems, while design 1 deploys available active systems and provides for additional resources, to be used for services in the case of disasters.
- The shared systems appear to be the most efficient use for the money.
- They have a component in common that manages the redundancy of the primary and the disaster-recovery system. This management component is almost always a single point of failure.
- In addition, shared systems replicate all data changes immediately between primary and disaster-recovery systems, and thus also replicate logical errors immediately.
- In modern architectures, virtualization technologies like domains or virtual hosts help us to segregate hardware and operating system resources and make them available for separate disaster cases.
- To get better usage of computer resources, cold-standby systems can be utilized as a test or a preproduction system, i.e., to analyse technical problems or to stage new releases.
- Within limits, this can also be done on hot-standby systems. There one needs the ability to turn off synchronization and turn it on again later without problems.

15. What can the hardware and software management tools be applied to? What do these tools address?

Ans.:

- Management tools can be software, hardware, a combination of the two, or be available as a service.
- A combination approach might involve a hardware diagnostic or monitoring tool that provides information to software running on a server or virtual machine or via a cloud SaaS capability.
- Tools can be purchased, rented or leased, borrowed, or found on the Web as a free or contribution shareware model.
- Software and management tools pertain to:
  - SaaS, Infrastructure as a Service (SaaS), and Platform as a Service (PaaS)
  - Physical, virtual, and cloud resources
  - Servers, storage, and networking hardware devices

- Application, middleware, and IRM software
- Operating systems and hypervisors
- Facilities, energy, power, cooling, and HVAC

- **Software and management tools address:**
- Application, business functionality, and information services
- Service and help desk, incident, and trouble tickets
- Performance, availability, capacity, and change tracking
- Data movement and migration
- IT services delivery and IRM activities
- Testing, simulation, diagnostics, and benchmarking
- Security, compliance, and identity management
- Data protection, including backup/restore and high availability (HA)
- Business continuance (BC) and disaster recovery (DR)
- Data footprint reduction (DFR) and optimization
- Planning and analysis, forecasting, acquisition, and disposition of IT resources
- Configuration, provisioning, troubleshooting, and diagnostics
- Metrics and measurement for enabling situational awareness
- Activity and event monitoring, accounting, and chargeback

16. Explain the different management tool interfaces.
Ans.:

- Management tools have interfaces leverage policies acting on data feeds from different sources such as event logs, usage, or activity to support automated operations.
- In following figure, element managers are shown for managing storage configuration, diagnostics, snapshots, and other device-specific and detailed functions.
- Also shown are SRM tools for monitoring and management and systems resource analysis (SRA) tools. These tools can also interface with each other and with framework tools.
- Figure 13.3 shows various management tools and interfaces involved in using cloud services.
- Some of the images shown in Figure 13.3 include a Web-based SaaS provider's health status display (right), a configuration applet running on a client's or user's computer (center) and a Web-accessible management console for IaaS.

- Various technologies include network access, bandwidth, and data footprint reduction optimization, HTTP and REST.
- Event and performance information can be downloaded via CSV or XML to be imported into other tools for analysis or event correlation.
- In addition to application, Web- or browser-based GUI and CLIs, other management interfaces for accessing or moving information include HTTP, FTP, XML, JSON, RSS, CSV, SOAP, and REST.
- Application program interfaces (APIs) and software development kits (SDKs) for vendor and industry, along with defector standards, include those from Amazon, GoGrid, Google, Open Stack, and Rackspace.
- Other management tools, interfaces, standards, and APIs include the Cloud Management Working Group (CMWG), Data Management Task Force (DMTF), Virtualization Management Initiative (VMAN), System Management Architecture for Server Hardware (SMASH), and Desktop and Mobile Architecture for Server Hardware (DASH).

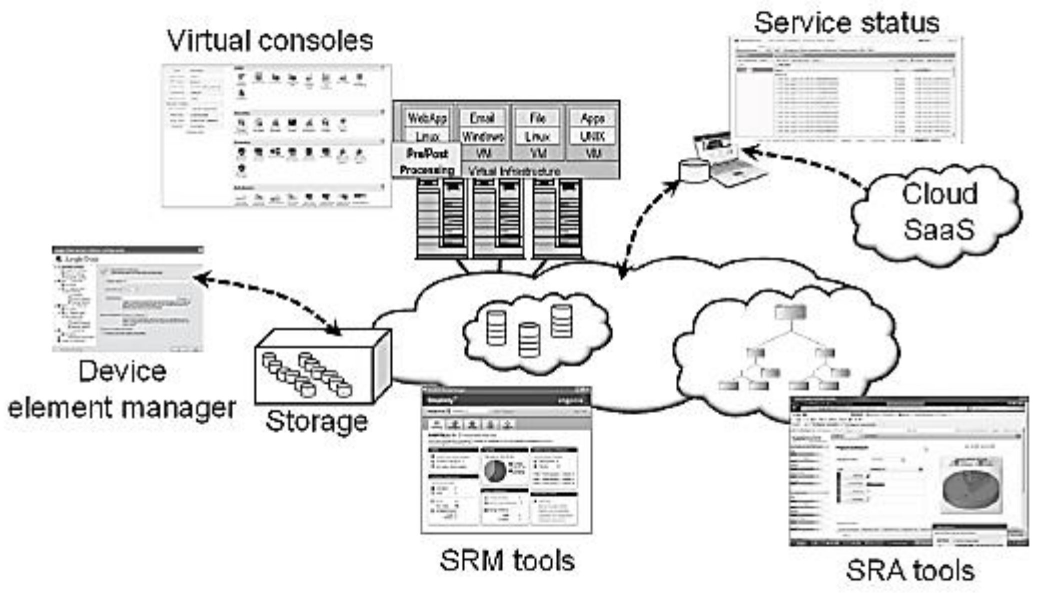


Fig: Management tool interfaces

- Others are the European Telecommunications Standards Institute (ETSI), Intel's Open Data Center Alliance, the National Institute of Standards and Technology (NIST), the Open Cloud Consortium (OCC), Open Cloud Computing Interface (OCCI), Open Grid Forum (OGF), Object Management Group (OMG), and Organization for the Advancement of Structured Information Standards (OASIS).
- Other initiatives and standards include Simple Network Management Protocol (SNMP), Management Information Bases (MIB), Storage

Networking Industry Association (SNIA) Storage Management Initiative Specification (SMIS), and Cloud Data Management Interface (CDMI).

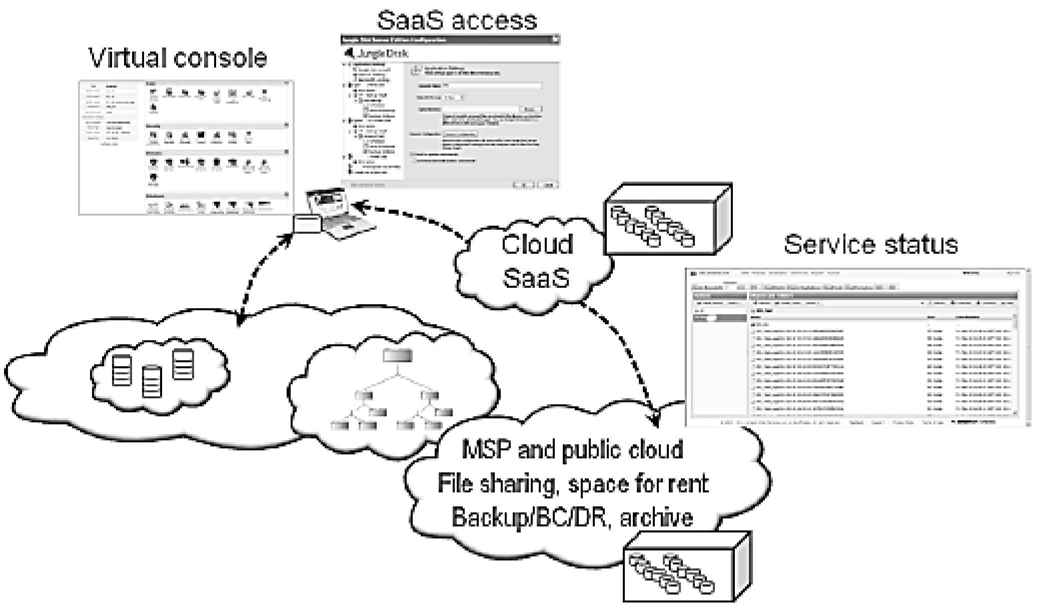


Fig: Cloud storage dashboards

17. What are the different ways of software and tools licensing? What are the considerations involved in software, tool, cloud and virtualization licensing?

Ans.:

- Software or tools may be licensed on a one-time, perpetual basis or on a time-duration basis.
- Software and tools can be licensed by
  - Flat fee per single or multiple copies, with or without support
  - Perpetual (no time limit) until you buy a new version
  - Limited time during which to use the software, get updates, and receive support
  - Commercial, education, government, or personal use
  - Enterprise or organization-wide and site-based
  - Individual applications, systems, servers, or workstations
  - Physical server or workstation instance
  - Per CPU (single, multicore, or socket-based), physical or virtual
  - Unlimited or on an hourly rate basis
  - Per user, seat, or workstation (total or concurrent in use)
  - Volume purchasing programs and tiered pricing
- Software, tool, cloud, and virtualization licensing consolidation involves the following considerations:

- Licensing for virtual environments should involve contracts and system administrators.
- Software licenses should be included as part of an IT resource capacity plan forecast process.
- When allocating VMs from templates, be sure there are provisional licenses available.
- Know how and when VMs are counted; is it an annual spot check or an average?
- Have a license review performed to ensure that you are getting the best ROI on those resources.
- Leverage tiered hypervisors from different vendors aligned to various service needs.
- Align the applicable tier of hypervisors and tools to the level of service needed.
- Per-VM licensing can be good fit for systems that do not lend themselves to consolidation.
- Per-physical-server licensing can be a good fit for applications that can be consolidated.
- If your organizations is concerned about hardware costs, then also focus on software fees

18. Explain hard and soft products.
Ans.:

- A soft product is the result of how various resources including hardware or software are delivered as a service.
- A hard product is some technology such as hardware (server, storage, network, or workstation), software, or service, including network bandwidth, which can be used or deployed in different ways.
- Soft product refers to what you do with the hard product technologies to deliver various services combining best practices, personalities, or functionalities via configuration, look and feel, and other forms of customization.
- The importance of understanding soft products is that what goes into making them has different costs or enables various services while using common components.
- Leveraging different metrics and measurements provides insight into how services are being delivered while reducing waste or rework that takes away from productivity while increasing costs.

- Maximizing resources is part of the objective for reducing costs, but so is reducing complexity, waste, rework, and other lost opportunities, including downtime or delays in enabling new services.
- Automation can help to address time-consuming and error-prone common processes such as analysis of event logs, resource discovery, service-level monitoring, and change validation.

19. What are the different issues in data center management? Explain.
Ans.:
The different issues in data center management is as follows:
• Making common users happy:
The data center should be designed to provide quality service to the majority of users for at least 30 years.
• Controlled information flow:
Information flow should be streamlined. Sustained services and high availability (HA) are the primary goals.
• Multiuser manageability:
The system must be managed to support all functions of a data center, including traffic flow, database updating, and server maintenance.
• Scalability to prepare for database growth:
The system should allow growth as workload increases. The storage, processing,
I/O, power, and cooling subsystems should be scalable.
• Reliability in virtualized infrastructure:
Failover, fault tolerance, and VM live migration should be integrated to enable recovery of critical applications from failures or disasters.
• Low cost to both users and providers:
The cost to users and providers of the cloud system built over the data centers should be reduced, including all operational costs.
• Security enforcement and data protection:
Data privacy and security defense mechanisms must be deployed to protect the data center against network attacks and system interrupts and to maintain data integrity from user abuses or network attacks.
• Green information technology:
Saving power consumption and upgrading energy efficiency are in high demand when designing and operating current and future data centers.

20. Discuss the challenges in developing cloud architecture.

Ans.:

- There are six challenges in cloud architecture development.

Challenge 1—Service Availability and Data Lock-in Problem:-

- The management of a cloud service by a single company is often the source of single points of failure. To achieve HA, one can consider using multiple cloud providers.
- Even if a company has multiple data centers, it may have common software infrastructure and accounting systems. Therefore, using multiple cloud providers may provide more protection from failures.
- Another availability obstacle is distributed denial of service (DDoS) attacks. Some utility computing services offer SaaS providers the opportunity to defend against DDoS attacks by using quick scale-ups.
- Software stacks have improved interoperability among different cloud platforms, but the APIs itself are still proprietary.
- Thus, customers cannot easily extract their data and programs from one site to run on another.
- The solution is to standardize the APIs so that a SaaS developer can deploy services and data across multiple cloud providers. This will rescue the loss of all data due to the failure of a single company.

Challenge 2—Data Privacy and Security Concerns:

- Current cloud offerings are essentially public (rather than private) networks, exposing the system to more attacks.
- Many obstacles can be overcome immediately with well-understood technologies such as encrypted storage, virtual LANs, and network middleboxes (e.g., firewalls, packet filters).
- For example, we could encrypt the data before placing it in a cloud.
- Traditional network attacks include buffer overflows, DoS attacks, spyware, malware, rootkits, Trojan horses, and worms.
- In a cloud environment, newer attacks may result from hypervisor malware, guest hopping and hijacking, or VM rootkits.
- Another type of attack is the man-in-the-middle attack for VM migrations.
- In general, passive attacks steal sensitive data or passwords. Active attacks may manipulate kernel data structures which will cause major damage to cloud servers.

Challenge 3—Unpredictable Performance and Bottlenecks:

- Multiple VMs can share CPUs and main memory in cloud computing, but I/O sharing is problematic. One solution is to improve I/O architectures and operating systems to efficiently virtualize interrupts and I/O channels.
- Internet applications continue to become more data-intensive. Cloud users and providers have to think about the implications of placement and traffic at every level of the system, if they want to minimize costs.
- This kind of reasoning can be seen in Amazon's development of its new CloudFront service.
- Therefore, data transfer bottlenecks must be removed, bottleneck links must be widened, and weak servers should be removed.

Challenge 4—Distributed Storage and Widespread Software Bugs:

- The database is always growing in cloud applications.
- Data centers must meet programmers' expectations in terms of scalability, data durability, and HA.
- Data consistence checking in SAN-connected data centers is a major challenge in cloud computing.
- Large-scale distributed bugs cannot be reproduced, so the debugging must occur at a scale in the production data centers.
- One solution may be a reliance on using VMs in cloud computing. The level of virtualization may make it possible to capture valuable information in ways that are impossible without using VMs.
- Debugging over simulators is another approach to attacking the problem, if the simulator is well designed.

Challenge 5—Cloud Scalability, Interoperability, and Standardization:

- The pay-as-you-go model applies to storage and network bandwidth; both are counted in terms of the number of bytes used.
- GAE automatically scales in response to load increases and decreases; users are charged by the cycles used.
- AWS charges by the hour for the number of VM instances used, even if the machine is idle.
- The opportunity here is to scale quickly up and down in response to load variation, in order to save money, but without violating SLAs.
- Open Virtualization Format (OVF) describes an open, secure, portable, efficient, and extensible format for the packaging and distribution of VMs.

- OVF also defines a transport mechanism for VM templates, and can apply to different virtualization platforms with different levels of virtualization.
- In terms of cloud standardization, we suggest the ability for virtual appliances to run on any virtual platform.
- We also need to enable VMs to run on heterogeneous hardware platform hypervisors. This requires hypervisoragnostic VMs.

Challenge 6—Software Licensing and Reputation Sharing:

- Many cloud computing providers originally relied on open source software because the licensing model for commercial software is not ideal for utility computing.
- The primary opportunity is either for open source to remain popular or simply for commercial software companies to change their licensing structure to better fit cloud computing.
- Another legal issue concerns the transfer of legal liability. Cloud providers want legal liability to remain with the customer, and vice versa.
- This problem must be solved at the SLA level.

21. Explain the structure of data-center networking.
Ans.:

- The core of a cloud is the server cluster (or VM cluster). Cluster nodes are used as compute nodes. A few control nodes are used to manage and monitor cloud activities.
- The scheduling of user jobs requires that you assign work to virtual clusters created for users.
- The gateway nodes provide the access points of the service from the outside world. These gateway nodes can be also used for security control of the entire cloud platform.
- Data-center server clusters are typically built with large number of servers, ranging from thousands to millions of servers (nodes).
- For example, Microsoft has a data center in the Chicago area that has 100,000 eight-core servers, housed in 50 containers.
- Data-center networks are mostly IP-based commodity networks, such as the 10 Gbps Ethernet network, which is optimized for Internet access.
- The figure shows a multilayer structure for accessing the Internet. The server racks are at the bottom Layer 2, and they are connected through fast switches (S) as the hardware core.

- The data center is connected to the Internet at Layer 3 with many access routers (ARs) and border routers (BRs).

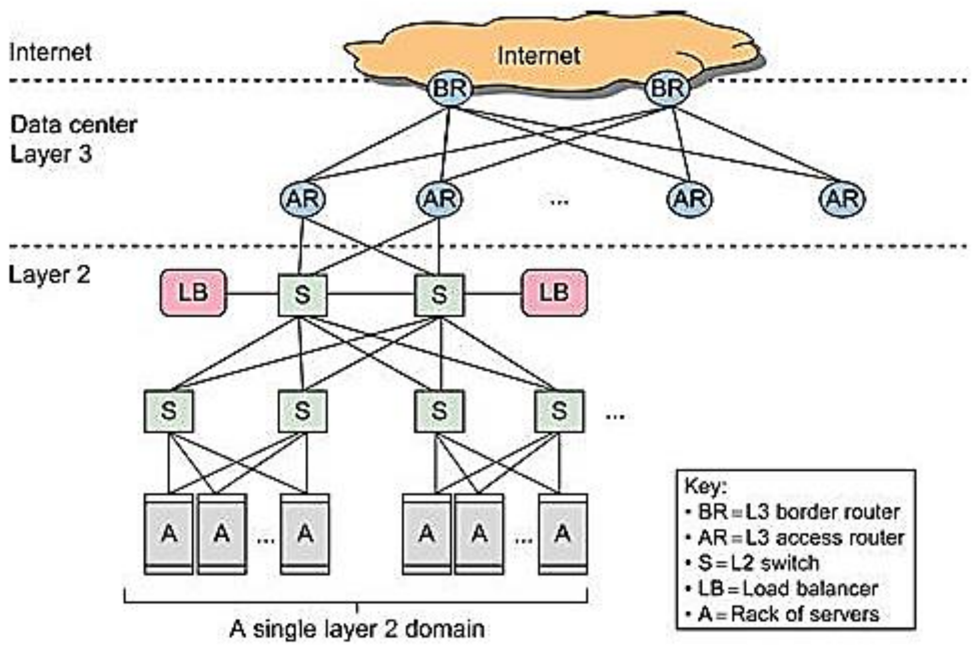


Fig: Standard data-center networking for the cloud to access the Internet

- An example of a private cloud is the one the U.S. National Aeronautics and Space Administration (NASA) is building to enable researchers to run climate models on remote systems it provides.
- This can save users the capital expense of HPC machines at local sites. Furthermore, NASA can build the complex weather models around its data centers, which is more cost-effective.
- These cloud models demand different levels of performance, data protection, and security enforcement. In this case, different SLAs may be applied to satisfy both providers and paid users.

22. Explain the generic cloud architecture.
Ans.:

- The following figure shows a security-aware cloud architecture.
- The Internet cloud is envisioned as a massive cluster of servers. These servers are provisioned on demand to perform collective web services or distributed applications using data-center resources.
- The cloud platform is formed dynamically by provisioning or de-provisioning servers, software, and database resources.

- Servers in the cloud can be physical machines or VMs. User interfaces are applied to request services. The provisioning tool carves out the cloud system to deliver the requested service.

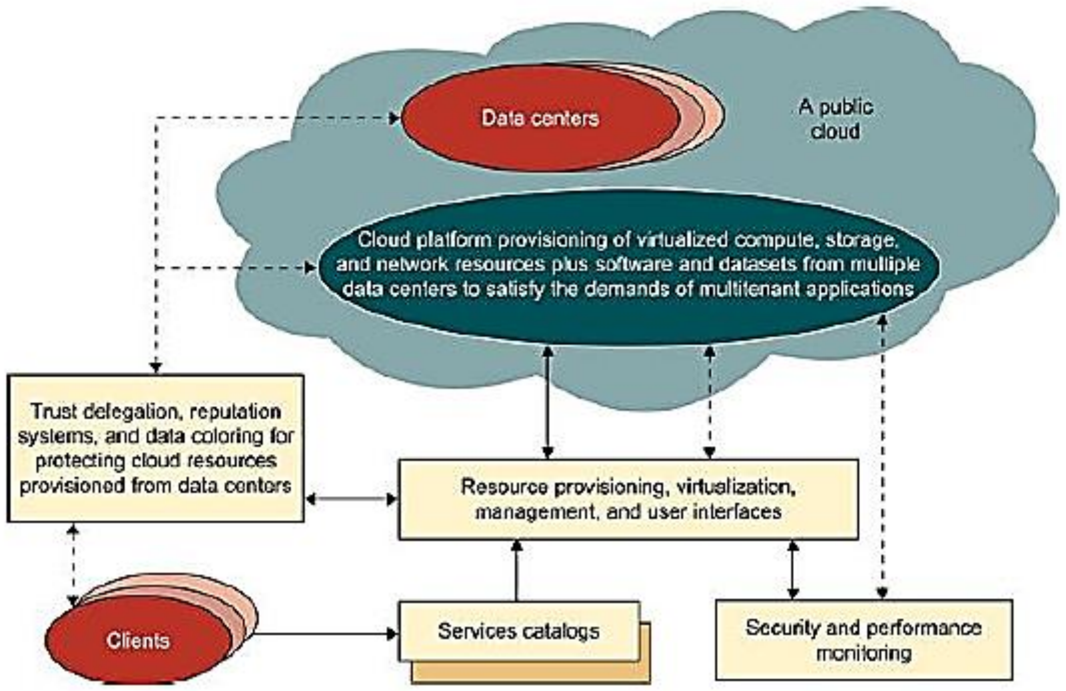

Data centers
A public cloud
Cloud platform provisioning of virtualized compute, storage, and network resources plus software and datasets from multiple data centers to satisfy the demands of multitenant applications
Trust delegation, reputation systems, and data coloring for protecting cloud resources provisioned from data centers
Resource provisioning, virtualization, management, and user interfaces
Clients
Services catalogs
Security and performance monitoring

Fig: A security-aware cloud platform built with a virtual cluster of VMs, storage, and networking resources over the data-center servers operated by providers

- In addition to building the server cluster, the cloud platform demands distributed storage and accompanying services.
- The cloud computing resources are built into the data centers, which are typically owned and operated by a third-party provider.
- In a cloud, software becomes a service. The cloud demands a high degree of trust of massive amounts of data retrieved from large data centers.
- We need to build a framework to process large-scale data stored in the storage system. This demands a distributed file system over the database system.
- Other cloud resources are added into a cloud platform, including storage area networks (SANs), database systems, firewalls, and security devices.
- Web service providers offer special APIs that enable developers to exploit Internet clouds. Monitoring and metering units are used to track the usage and performance of provisioned resources.
- The software infrastructure of a cloud platform must handle all resource management and do most of the maintenance automatically.

- Software must detect the status of each node server joining and leaving, and perform relevant tasks accordingly.
- Cloud computing providers, such as Google and Microsoft, have built a large number of data centers all over the world. Each data center may have thousands of servers.
- The location of the data center is chosen to reduce power and cooling costs. Thus, the data centers are often built around hydroelectric power.

23. Explain the layered architectural development of cloud platform for IaaS, PaaS and SaaS.

Ans.:

- Cloud computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers.
- The services provided over the cloud can be generally categorized into three different service models: namely IaaS, Platform as a Service (PaaS), and Software as a Service (SaaS).
- The following figure illustrates three cloud models at different service levels of the cloud.
- SaaS is applied at the application end using special interfaces by users or clients.
- At the PaaS layer, the cloud platform must perform billing services and handle job queuing, launching, and monitoring services.
- At the bottom layer of the IaaS services, databases, compute instances, the file system, and storage must be provisioned to satisfy user demands.

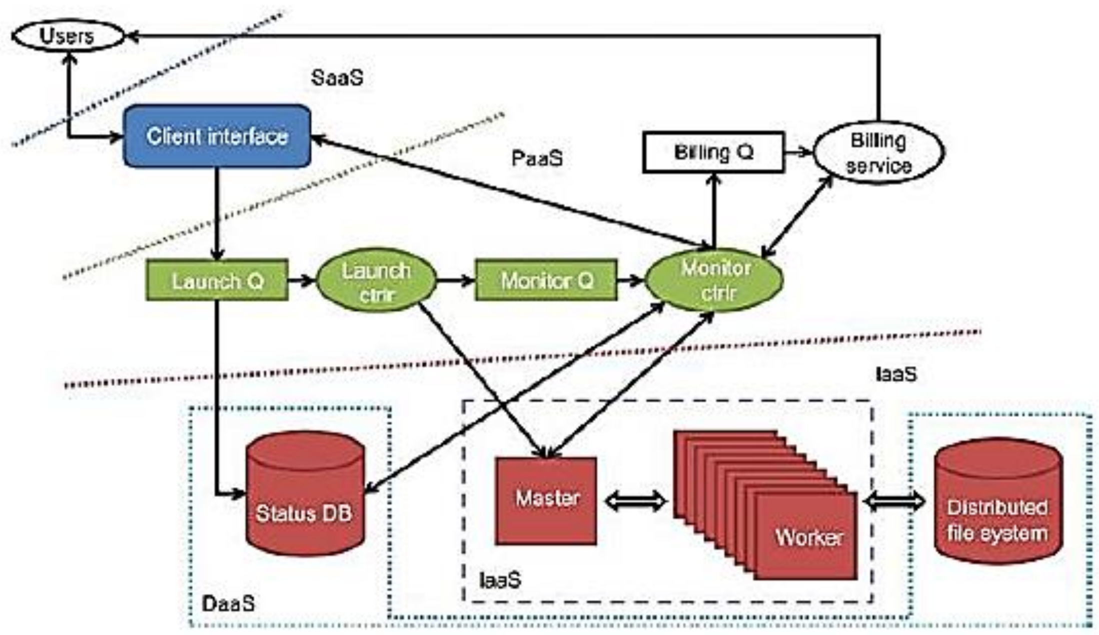


Fig: The IaaS, PaaS, and SaaS cloud service models at different service levels.

❖ **Infrastructure as a Service:**

- This model allows users to use virtualized IT resources for computing, storage, and networking.
- In short, the service is performed by rented cloud infrastructure. The user can deploy and run his applications over his chosen OS environment.
- The user does not manage or control the underlying cloud infrastructure, but has control over the OS, storage, deployed applications, and possibly select networking components.
- This IaaS model encompasses storage as a service, compute instances as a service, and communication as a service.
- Amazon EC2, GoGrid, FlexiScale, and Aneka are good examples.

❖ **Platform as a Service (PaaS)**

- To be able to develop, deploy, and manage the execution of applications using provisioned resources demands a cloud platform with the proper software environment. Such a platform includes operating system and runtime library support.
- This has triggered the creation of the PaaS model to enable users to develop and deploy their user applications.
- The platform cloud is an integrated computer system consisting of both hardware and software infrastructure.
- The user application can be developed on this virtualized cloud platform using some programming languages and software tools supported by the provider (e.g., Java, Python, .NET).

- The user does not manage the underlying cloud infrastructure.
- The cloud provider supports user application development and testing on a well-defined service platform.
- This PaaS model enables a collaborated software development platform for users from different parts of the world.
- This model also encourages third parties to provide software management, integration, and service monitoring solutions.
- Google App Engine, Microsoft Azure, Amazon Elastic MapReduce, Aneka are good examples of PaaS.

❖ **Software as a Service (SaaS):**

- This refers to browser-initiated application software over thousands of cloud customers. Services and tools offered by PaaS are utilized in construction of applications and management of their deployment on resources offered by IaaS providers.
- The SaaS model provides software applications as a service. As a result, on the customer side, there is no upfront investment in servers or software licensing.
- On the provider side, costs are kept rather low, compared with conventional hosting of user applications.
- Customer data is stored in the cloud that is either vendor proprietary or publicly hosted to support PaaS and IaaS.
- The best examples of SaaS services include Google Gmail and docs, Microsoft SharePoint, and the CRM software from Salesforce.com.
- They are all very successful in promoting their own business or are used by thousands of small businesses in their day-to-day operations.

24. Explain the market oriented cloud architecture.
Ans.:

- As consumers rely on cloud providers to meet more of their computing needs, they will require a specific level of QoS to be maintained by their providers, in order to meet their objectives and sustain their operations.
- To achieve this, the providers cannot deploy traditional system-centric resource management architecture.
- Instead, market-oriented resource management is necessary to regulate the supply and demand of cloud resources to achieve market equilibrium between supply and demand.
- The purpose is to promote QoS-based resource allocation mechanisms. In addition, clients can benefit from the potential cost reduction of

providers, which could lead to a more competitive market, and thus lower prices.

- Figure shows the high-level architecture for supporting market-oriented resource allocation in a cloud computing environment.

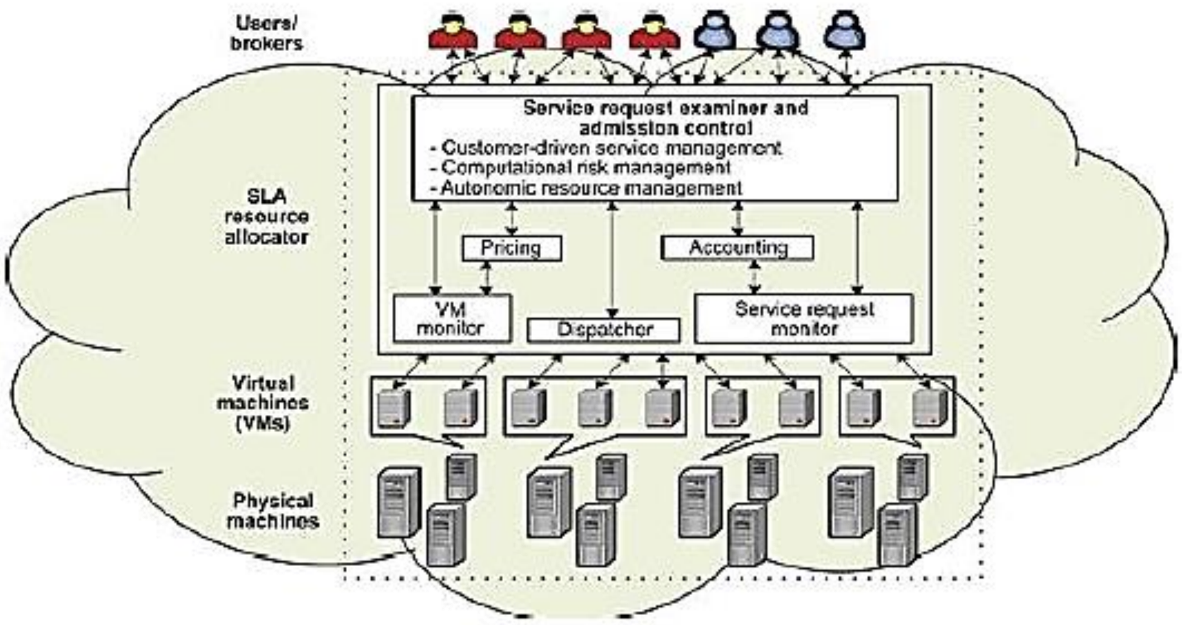

Users/ brokers
SLA resource allocator
Service request examiner and admission control
- Customer-driven service management
- Computational risk management
- Autonomic resource management
Pricing
Accounting
VM monitor
Dispatcher
Service request monitor
Virtual machines (VMs)
Physical machines

Fig: Market-oriented cloud architecture to expand/shrink leasing of resources with variation in QoS/demand from users

- This cloud is basically built with the following entities:
- ➢ **Users or brokers** acting on user's behalf submit service requests from anywhere in the world to the data center and cloud to be processed.
- ➢ The **SLA resource allocator** acts as the interface between the data center/cloud service provider and external users/brokers. It requires the interaction of the following mechanisms to support SLA-oriented resource management.
- ➢ The **request examiner** ensures that there is no overloading of resources whereby many service requests cannot be fulfilled successfully due to limited resources.
- ➢ It also needs the latest status information regarding resource availability and workload processing in order to make resource allocation decisions effectively. Then it assigns requests to **VMs** and determines resource entitlements for allocated VMs.
- ➢ The **Pricing** mechanism decides how service requests are charged. Pricing serves as a basis for managing the supply and demand of computing resources within the data center and facilitates in prioritizing resource allocations effectively.

- The **Accounting** mechanism maintains the actual usage of resources by requests so that the final cost can be computed and charged to users.
- The **VM Monitor** mechanism keeps track of the availability of VMs and their resource entitlements.
- The **Dispatcher** mechanism starts the execution of accepted service requests on allocated VMs.
- The **Service Request Monitor** mechanism keeps track of the execution progress of service requests.
- Multiple VMs can be started and stopped on demand on a single physical machine to meet accepted service requests, hence providing maximum flexibility to configure various partitions of resources on the same physical machine to different specific requirements of service requests.