S. No.	Date	Title	Page No.	Teacher's Sign / Remarks

UNIT-1

common Q's:

- PaaS, SaaS, IaaS (comp)
- AWS - storage services
- RAID-5 tech.
- NIST cloud ref. model
- Etc., OpenN & Nimbus (comp)

UNIT-2

- Life cycle of a workflow - phases
- workflow patterns
- Map-reduce philos.
- workflow & programs
- Grep, Zookeeper

UNIT-1

NETWORK-CENTRIC COMPUTING & NETWORK-CENTRIC CONTENT SHARED CHARACTERISTICS

* Most applications are data-intensive. Computer simulation becomes a powerful tool for scientific research in virtually all areas of science. Tools for CAD are widely used in the aerospace & automotive industry. The widespread use of sensors contributes to increases in the volume of data. Multimedia app^{n}s are popular - media increases the load placed on storage, networking, & processing systems.

* Virtually all app^{n}s are network-intensive. Transferring large volumes of data requires high-bandwidth networks; parallel computing, computation steering (guide a computational experiment toward a region of interest), & data streaming are examples of app^{n}s that can only run efficiently on low-latency (delay) networks.

* The systems are accessed using thin clients running on systems w/ limited resources. (end-point users)

* The infrastructure supports some form of workflow management.

ADVANTAGES / SOURCES OF CONCERN (NCComp & NCCont.)

- computing & communication resources are shared & resources can be aggregated to support data-intensive app's. Multiplexing leads to a higher resource utilization; when multiple ~~systems~~ app's share a system, their peak demands for resources are not synchronized. The management of large pools of resources poses challenges as complex systems are subject to phase transitions. Ensuring QoS guarantees is challenging in such environments as total performance isolation is elusive. (diff. to achieve)

- Data sharing facilitates collaborative activities & projects. This poses not only security & privacy challenges but also requires mechanisms for access control by authorized users & for detailed logs of the history of data changes.

- Cost reduction. Concentration of resources creates the opportunity to pay as you go for computing & thus eliminates the initial investment & significantly reduces the maintenance & operation costs.

- User convenience & elasticity, that is the ability to accommodate workloads with very large peak-to-average ratios.

PEER-TO-PEER SYSTEMS

Desirable properties:

* They require a minimally dedicated infrastructure, since resources are contributed by the participating systems.
* They are highly decentralized.
* " " scalable
* " " resilient to faults & attacks, since few of their elements are critical for the delivery of service & the abundance of resources can support replication.
* Individual nodes do not require excessive network bandwidth the way servers used in case of the client-server model do.
* Systems are shielded from censorship due to the dynamic & often unstructured system architecture.

Undesirable Properties

* Decentralization raises the question of whether P2P systems can be managed efficiently & provide the security required by various app^n^s.
* The fact that they are shielded from censorship makes them a fertile ground for illegal activities.

→ Nodes w/ abundant resources in systems w/out any centralized infrastructure often act as supernodes & maintain info useful to increasing the system efficiency.

→ Regardless of the architecture, P2P systems are built around an overlay network, a virtual network superimposed over the real network.

→ Each node maintains a table of overlay links connecting it w/ other nodes of this virtual network.

Overlay networks → structured
                 → unstructured

CLOUD COMPUTING:

* Large data centers are more efficient than medium-sized data centers. because:

→ They are more economical to operate

→ Large d.c equipped w/ commodity computers experience a 5-7 times decrease of resource consumption, including energy.

→ The networking costs are 7.1 times larger, & the storage costs are 5.7 times larger for med-sized centers.

→ Med-sized data centers have a larger administrative overhead.

Types of Cloud:

- Private cloud - The infrastructure is operated solely for an organization.
- Community " - " " is shared by several organizations & supports a specific community that has shared concerns.
- Public " - " " is made available to the general public or a large industry group & is owned by an org. selling clouds.
- Hybrid cloud - " " is a composition of 2 or more clouds (priv., comm., or pub.) that remain unique entities but are bound together by standardized or proprietary technology that enables data & application portability.

Reasons for the success of cloud computing

* CC is in a better position to exploit recent advances in software, networking, storage, & processor technologies.
* A cloud consists of a homogeneous set of hardware & software resources in a single administrative domain. In this setup, security, fault tolerance, resource management & QoS are less challenging than in a heterogeneous environment.
* CC is focused on enterprise computing.
* A cloud provides the illusion of infinite computing resources.
* A cloud eliminates the need for up-front financial commitment, & it is based on a pay-as-you-go approach.

Obstacles

- Availability of service
- Vendor lock-in
- Data confidentiality & auditability
- Data transfer bottlenecks
- Performance unpredictability
- Elasticity, the ability to scale up & down quickly

CC DELIVERY MODELS & SERVICES

Entities involved in CC :

* Service Consumer - the entity that maintains a business relationship w/ & uses services from service providers
* Service provider - " " responsible for making a service available to service consumers.
* Carrier - The intermediary that provides connectivity & transport of cloud services b/w prov. & cons.
* Broker - " " that manages the use, perf., & delivery of cloud services & negotiates relationships b/w cons. & prov.
* Auditor - A party that can conduct independent assesment of cloud services, info system operations, perf. & security of the cloud implementation.
- Audit - It is a systematic evaluation of a cloud system that measures how well it conforms to a set of established criteria.

Delivery models — SaaS, PaaS & IaaS

① Software-as-a-Service (SaaS)

* Gives the capability to use applications supplied by the service provider in a cloud infrastructure
* The app^{n}s are accesible from various client devices through a thin-client interface (web browser)
* The user doesn't manage or control the underlying cloud infrastructure, w/ the possible exception of limited user-specific app^n configuration settings.

Services offered

* Enterprise services such as workflow management, groupware & collaborative, supply chain, communications, digital signature, customer rel^{n}. management (CRM), desktop S/W, financial management, geo-spatial & search.
* Web 2.0 app^{n}'s such as metadata management, social networking, blogs, wiki services & portal services.

* The SaaS is not suitable for app^{n}'s that require real-time response or those for which data is not allowed to be hosted externally.

② Platform-as-a-service (PaaS)

* Gives the capabilities to deploy consumer-created, app^{n}'s using (or acquired) programming languages & tools supported by the provider.
* The user doesn't manage or control the underlying cloud infrastructure.
* The user has control over the deployed app^{n}'s & possibly, over the app^{n} hosting environment configurations. Such

services include session management, device integration, sandboxes, instrumentation & testing, contents management, knowledge management, UDDI (Univ. Descr. Integr. Disc.), a platform-independent XML-based registry.

* PaaS is not useful when the app^{n} must be portable, * when proprietary programming languages are used, or * when the underlaying hardware & software must be customized to improve the performance of the app^{n}.

③ Infrastructure-as-a-Service

* Capability to provision processing, storage, networks, & other fundamental computing resources.
* The consumer is able to deploy & run arbitrary S/W, which can include OS & app's.

* The consumer doesn't manage or control the underlying cloud infrastructure but has control over OS, storage, deployed app"s, & possibly limited control of some networking components.

* **Services Offered**

- Server hosting
- Web servers
- Computing hardware
- Load balancing
- Bandwidth Provisioning
- Storage
- OS
- Virtual instances
- Internet Access

* This model is particularly useful when the demand is volatile & a new business needs computing resources & doesn't want to invest in a computing infrastructure or when an org is expanding rapidly.

Activities necessary to support the delivery models

* Service management & provisioning, including virtualization, call center, operations mgt, sytems mgt, QoS mgt, billing & accounting, asset mgt, technical support, & backups.

* Security mgt., including ID & authentic -ation, certification & accreditation, intrusion prevention, intrusion detection, virus protection, cryptography, access control, firewalls.

* Customer services such as customer assistance & online help, subscriptions, busines intelligence, customer preferances & personalization

* Integration services, including data mgt & development.

ETHICAL ISSUES IN CC

* CC is based on a paradigm shift. main elements of this shift:
  - the control is relinquished to third-party services
  - the data is stored on multiple sites administered by several organizations
  - multiple services interoperate across the network

* Unauthorized access, data corruption, infrastructure failure, & service unavailability are some risks related to relinquishing control; moreover, when a problem occurs, it is difficult to identify the source.

* Systems can span the boundaries of multiple organizations & cross security boundaries, a process called deperimeterization.

* Unlimited data sharing test the right or ability of individuals to exercise personal control over the collection, & the disclosure of their personal data by others.

** Accountability is a necessary ingrediant of CC; adequate info. about how data is handled w/in the cloud & about allocation of responsibility are key elements for enforcing ethics rules in CC.

CLOUD VULNERABILITIES

* Malicious attacks & failure of the infrastructure

* Stability risks due to interacting services

** Clustering the resources in data centers located in diff. geographical areas is one of the means used to lower the probability of catastrophic failures.

CHALLENGES FACED BY CC

* security
* Resource Mgt. → controllers to implement several classes of policies: admission control, capacity allocation, energy optimization & to provide QoS guarantees.
* Interoperability & standardization

CLOUD INFRASTRUCTURE

Amazon Web Services (AWS)

Storage services

① Simple Storage System (S3):

* Designed to store large objects
* Supports a minimal set of functions: write, read & delete.
* Allows an app^n to handle an unlimited no. of objects ranging in size from 1 byte to 5 terabytes.
* An object is stored in a bucket & retrieved via a unique developer-assigned key.
* Maintains the name, modification time, an access control list, & up to 4 kilobytes of user-defined metadata for each object.
* Object names are global
* Authentication mechanisms ensure that data is kept secure.
* Objects can be made public
* S3 supports PUT, GET & DELETE primitives to manipulate objects, but doesn't support prim. to copy, rename or move an object to another bucket.
* S3 computes the MD5 (cryptographic hash function) of every object & returns it in the field ETag.
  A user has to compute the MD5 of an object stored & compare it w/ the ETag; if the 2 values don't match, the object was corrupted during transmission or storage.
* S3 uses standards-based REST and SOAP interfaces; the default download protocol is HTTP, but BitTorrent protocol interface is also provided to lower costs for high-scale distribution.

② Elastic Block Store (EBS):

* Provides persistent block-level storage volumes for use w/ Amazon EC2 instances.

* A volume appears to an app^n as a raw, unformatted, & reliable physical disk.

* Size of the volume ranges from 1 gigabyte to 1 terabyte

* The volumes are grouped together in availability zones & are automatically replicated in each zone.

* An EC2 instance may mount multiple volumes, but a volume cannot be shared by multiple instances.

* EBS supports the creation of snapshots of the volumes attached to an instance & uses them to restart an instance.

* The storage strategy provided by EBS is suitable for database app^{n}s, file systems, & app^{n}s using raw data devices.

③ Simple DB

* Non-relational data store

* Allows developers to store & query data items via web services requests.

* Supports store-and-query functions

* Creates multiple geographically distributed copies of each data item & supports high-performance web app^{n}s.

* Automatically manages infrastructure provisioning, hardware & software maintenance, replication & indexing of data, & performance tuning.

* Elastic Compute Cloud (EC2)

* A web service w/ a simple interface for launching instances of an app^n under several OS's.

* An instance is created either from a predefined AMI (Amazon Machine Image) digitally signed & stored in S3 or from a user-defined image.

* The image includes the OS, the run-time environment, the libraries & the app^n desired by the user.

* AMI images make an exact copy of the original image but without configuration-dependent info. such as hostname.

* A user can:
  - Launch an instance from an existing AMI & terminate an instance
  - Start & stop an instance
  - Create a new image
  - Add tags to identify an image
  - Reboot an instance

* EC2 is based on the Xen virtualization strategy.

* In EC2, each virtual machine or instance functions as a virtual private server.

* An instance specifies the max amt. of resources available to an app^n, the interface for that instance, & the cost per hour.

* A user can interact w/ EC2 using a set of SOAP msgs.

* The user has root access to each instance in the elastic & secure computing environment of EC2.

* The instances can be placed in multiple locations in different regions & availability zones.

* (EC2) It also allows the import of virtual machine images from the user env. to an instance through a facility called VM import.

Simple Queue Service (SQS)

* Hosted message queue
* It is a system for supporting automated workflows; it allows multiple EC2 instances to coordinate their activities by sending & receiving SQS messages.
* Any comp. connected to the Internet can add or read msgs w/out any installed software.
* App^n's using SQS can run independently & asynchronously & don't need to be developed w/ the same technologies.
* A received msg is locked during procesing; if procesing fails, the lock expires & the msg is available again.
* The time-out for locking can be changed dynamically via the ChangeMessageVisibility operation.
* Developers can access SQS through standards-based SOAP & Query interfaces.
* Queues can be shared w/ other AWS accounts.
* Queue-sharing can also be restricted by IP address & time-of-day.

CloudWatch

* Monitoring infrastructure
* Used by app^n developers, users & system administrators to collect & track metrics important for optimizing the performance of app^n's & for increasing the efficiency of resource utilization.
* W/out installing any software, a user can monitor approx a dozen preselected metrics & then view graphs & stats for these metrics.
* When launching an AMI, a user can start the CloudWatch & specify the type of monitoring.

* Basic Monitoring - Free of charge & collects data at 5-min intervals for up to 10 metrics.
* Detailed Monitoring - subject to charge & collects data at 1-min intervals.

* This service can also be used to monitor the latency of acces to EBS volumes, the available storage space for RDS DB instances, no. of messages in SQS, & other parameters.

Virtual Private Cloud
VPC

* Provides a bridge b/w the existing IT infrastructure & the AWS cloud.
* The connection is made via a Virtual Private Network (VPN).
* Allows existing management capabilities such as security services, firewalls & intrusion detection systems to operate seamlessly w/in the cloud.

AutoScaling

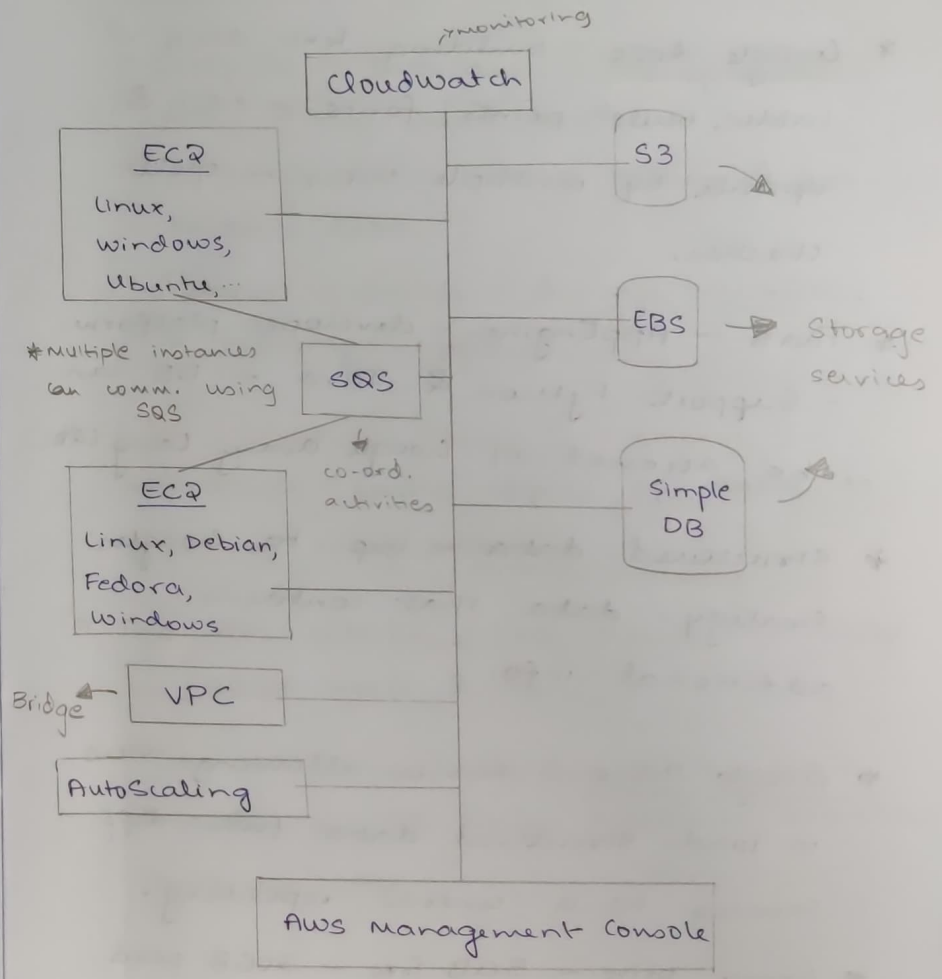
* Provides automatic scaling of EC2 instances.
* Supports grouping of instances, monitoring instances in a group, & defining triggers & pairs of CloudWatch alarms & policies, which allow the size of the group to be scaled up or down.
* Triggers use CloudWatch alarms to detect events & then initiate specific actions.

Elastic Beanstalk

* Interacts w/ other AWS services, including EC2, S3, SNS, Elastic Load Balance & Auto Scaling.
* Automatically scales the resources as required by the app^n.
* Mgt. functions provided:
  - deployment of a new app^n version
  - access to the results reported by Cloudwatch
  - access to server login files w/out needing to login to the app^n servers.
  - email notifs. when app^n status changes

Cloud Formation

* Allows the creation of a stack describing the infrastructure for an app^n.
* The user creates a template, a text file formatted as in JSON, describing the resources, the config. values, & the interconnection among these resources.



GOOGLE

* Their efforts are concentrated in the area of SaaS

* Data for services such as Gmail, Google Drive, Google Calendar & Google Groups is stored in data centers on the cloud.

* Google docs - building text docs - tables, bullet points, fonts - edit & updata by multiple users - spell checker.

* PaaS - AppEngine - developer platform - Supports Python & Java - DB can be accesed w/ Google Query Lang. (GQL)

* Structured data - imp. to Google's strategy - data that contains additional info.

* Google Base - servic allowing users to load structured data from diff sources to a central repository.

* Google Drive - 5GB free - 20GB paid.

Microsoft - Azure & Online Services

PaaS ↓ SaaS ↓

* Windows Azure - OS

  SQL Azure - cloud-based version of the SQL Server

  Azure AppFabric - collection of services for cloud app^n's.

* Core Components of Azure:
  - Compute - provides a computation env. Runs cloud app^n's.
  - Storage - uses blobs, tables & queues to store data
  - Fabric Controller - deploys, monitors & manages app^n's.

* CDN (Computer Delivery Network) - maintains cache copies of data to speed up computations.

* The API interface to Azure is built on REST, HTTP, & XML.

* The platform includes 5 services - Live Services, SQL Azure, AppFabric, SharePoint, Dynamics CRM.

* The computations carried out by an appⁿ are implemented as 1 or more roles; an appⁿ typically runs multiple instances of a role.

* Scaling, load balancing & reliability are ensured by the fabric controller. It also decides where new appⁿs should run.

* A blob contains binary data
* A container consists of 1 or more blobs
* The Micr. Azure platform doesn't support any distributed parallel computing frameworks.

OPEN SOURCE SOFTWARE PLATFORMS FOR PRIVATE CLOUDS

Open source cc platforms such as Eucalyptus, OpenNebula, & Nimbus can be used as a control infrastructure for a private cloud.

A cloud infrastructure carries out the following steps to run an appⁿ.

- Retrieves the user i/p from the front end
- Retrieves the disk image of a VM from a repository
- Locates a system & requests the VMM (virtual machine monitor) running on that system to set up a VM.
- Invokes the DHCP & the IP bridging software to set up a MAC & IP address for the VM.

① Eucalyptus

Components of the system:

- VM - runs under several VMMs
- Node Controller - Runs on every server designed to host a VM & controls the activities of the node.
- Cluster Controller - Controls servers. Interacts w/ the node controller to schedule requests on that node.
- Cloud Controller - Provides cloud access to users, dev.s & administrators.

- Storage Controller - provides persistent virtual hard drives to app^{n}s. It is the correspondent of EBS.
- Storage service - Provides persistent storage & allows users to store objects in buckets.

* The system supports a decentralized resource management of multiple clusters w/ multiple cluster controllers, but a single head node for handling user interfaces.

② Open-Nebula

* Private cloud w/ users actually logging into the head node to access cloud functions.
* System is centralized
* " is best suited for an operation involving a small-to-medium-sized group of trusted & knowledgeable users.

③ Nimbus

* cloud soln for scientific app^{n}s based on the Globus software.
* Inherits from Globus the image storage, credentials for user authentication, & the requirement that a running Nimbus process can ssh into all compute nodes.
* ssh - Network protocol that allows data to be exchanged using a secure channel b/w 2 networked devices.

	Eucalyptus	OpenNebula	Nimbus
Design	Emulate EC2	Customiza-ble	Based on Globus
Cloud Type	Private	Private	Public/Private
User Population	Large	Small	Large
Applications	All	All	Scientific
Internal security	Strict	Loose	Strict
User Access	user credentials	User credentials	x509 credentials
Network Access	To cluster controller	—	To each comp. node
Customiza-bility	Administrators & limited users.	Admins & users	All but image storage & credentials

CLOUD STORAGE DIVERSITY & VENDOR LOCK-IN

* RAID-5 system

SERVICE & COMPLIANCE LEVEL AGREEMENTS

* An SLA is a negotiated contract b/w the customer & the service provider.
* Objectives of this agreement:
  - Identify & define customers' needs & constraints
  - Provide a framework for understanding (Defn. of service & costs)
  - Simplify complex issues
  - Reduce areas of conflict
  - Encourage dialogue in the event of dispute
  - Eliminate unrealistic expectations.

* An SLA records a common understanding in areas like - services, priorities, responsibilities, guarantees, warranties.

* Phases in SLA management
  - Negot. of the contract
  - Monitoring its fulfillment in real time

UNIT-2

EXISTING CLOUD APPLICATIONS

Processing pipelines
Batch processing systems
web app^n's

* Processing pipelines are data-intensive & sometimes compute-intensive app^n's & represent a fairly large segment of app^n's currently running on the cloud.

DATA PROCESSING APP^{N}s

* Indexing – the processing pipeline supports indexing of large datasets created by web Crawler Engines.
* Data Mining – " " searching very large collections of records to locate items of interest.
* Image Processing – " " image conversion & compression or encryption of images
* Video Transcoding – Pp transcodes from 1 video format to another.
* Doc. processing – Pp converts very large docs from 1 format to another

BATCH PROCESSING APP^{N}s

* Generation of daily, weekly, monthly & annual activity reports for orgs.
* Processing, aggregation & summaries of daily transactions for financial institutions & other companies.
* Inventory management for large corps.
* Processing billing & payment records.
* Mgt of software development.
* Automatic testing & verification of software & hardware systems

NEW APP^N OPPORTUNITIES

* Batch proc. for decision support & other aspects of Business Analytics
* Parallel Batch processing based on programming abstractions.
* Mobile interactive app^{n}s that process large volumes of data from diff. sensors & services that combine more than 1 data source.

ARCHITECTURAL STYLES FOR CLOUD APP^{N}s

* Comm → clients & stateless servers
* Stateless Servers don't require a client to first establish a connection to the server. It views a client request as an independent transaction & responds.
* In the case of a st. serv, a client isn't affected while a server goes down & then comes back up b/w 2 consecutive requests. (usually, server failure → considerable overhead)

* A stateless system is simpler, more robust, & scalable.

* A client doesn't have to be concerned w/ the state of the server.

** Critical aspect of the dev. of networked app's → how processes & threads running on systems w/ diff. architectures & possibly compiled from diff programming langs can comm. structured info. w/ one another.

* (↑) The intanal representation of the 2 structures at the 2 sites may be different. (Big-Endian, Little-Endian)

* A comm. channel transmits bits & bytes; ∴ the data structure must be serialized at the sending site & reconstructed at the receiving site.

* Neutrality - Ability of the app^n protocol to use diff. transport protocols (TCP, UDP, ...) & to run on top of a diff. protowl stack.

* Extensibility - " to incorporate additional functions, like security.

* Independence - " to accomodate diff programming styles.

* Often the app^n clients & the servers running on the cloud comm. using RPCs

* Remote Procedure Call (RPC) - A protocol that a program can use to request a service from a program located in another computer on a network w/out having to understand the network's details.

* RPC-based app's use stubs (proxy func.) → marshalling the ds's, serialization

* Object Request Broker (ORB) - the middleware that facilitates comm. of networked app's.

ORB at Sending site
- Transforms d.s's used internally by the sending process to a byte sequence

transm. over the network

ORB at Receiving site
- Maps the byte seq to the d.s's " by the rec. process.

* CORBA (Common Object Request Broker Architecture) allows networked app^n's dev. in diff. programming lang, & running on systems w/ diff architectures & system software to work w/ one another.

* (↑) IDL (Interface Defn. lang) — used to specify the interface of an object

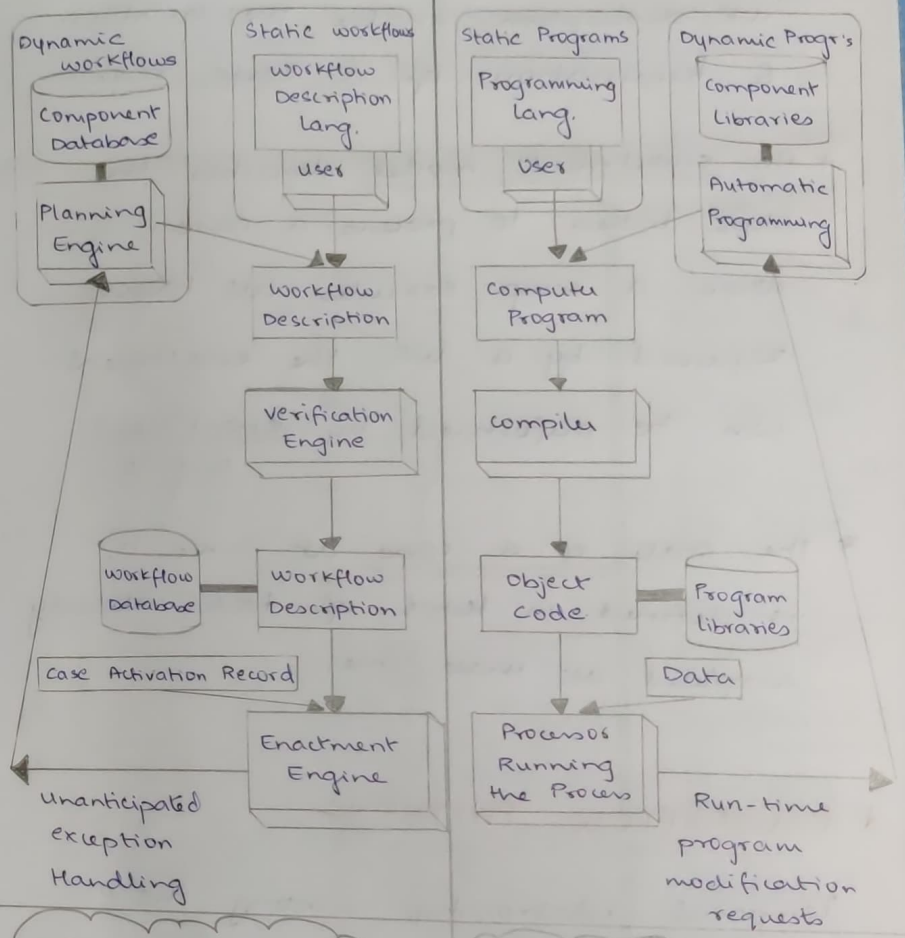
* SOAP (Simple Object Access Protocol) — dev for web app^n's; its message format is based on XML. SOAP uses TCP & UDP transport protocols. It can be stacked over other app^n layer protocols (HTTP, SMTP)

* WSDL (Web Services Description Lang) — XML-based grammar to describe comm. b/w endpoints of a networked app^n.

* REST (Representational State Transfer) — style of s/ware arch. for distributed hypermedia systems.

** * WORKFLOWS

A parallel b/w workflows & programs



Life cycle of a workflow

Phases:

Creation, Definition, Verification & Enactment

Life cycle of a computer program

Phases:

Creation, Compilation, & Execution

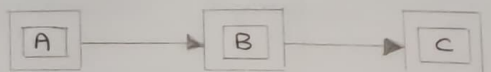
* A case is an instance of a process description
* The start & stop symbols in the WF description enable the creation & termination of a case, resp.
* An enactment model describes the steps taken to process a case. When a comp. executes all tasks required by a WF, the enactment can be performed by enactment engine.
* The state of a case at time t, is defined in terms of tasks already completed at that time.

** WORKFLOW PATTERNS

Temporal relationship among the tasks of a process.

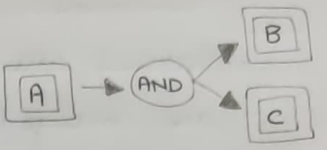
① SEQUENCE PATTERN

* Occurs when several tasks have to be scheduled 1 after the completion of the other.



② AND Split 'P

Requires several tasks to be executed concurrently.



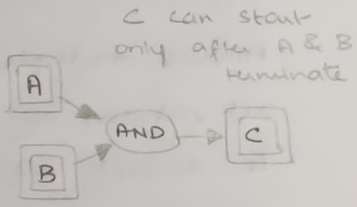
* Both tasks C & B are activated when A terminates.

Explicit AND split → Has a routing node w/ all activities connected to it (↑) that are activated as soon as the flow of control reaches the routing node.

Implicit AND split → Conditions can be associated w/ branches. Activities are activated only if the conditions are true for that branch.

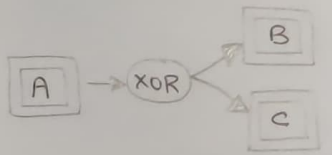
③ SYNCHRONIZATION 'P

Requires several concurrent activities to terminate before an activity can start.



④ XOR split

Requires a decision.



After completion of A, either B or C can be activated

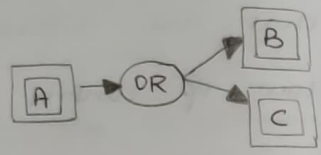
⑤ XOR join

Several alternatives are merged into one.

C is enabled when either A or B terminates

⑥ OR split

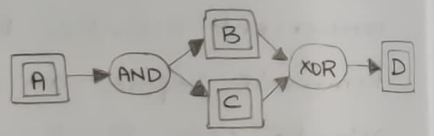
To choose multiple alternatives out of a set.



After completion of A, one could activate either B or C, or both.

⑦ Multiple Merge

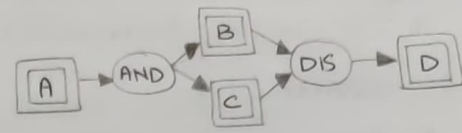
Allows multiple activations of a task, & doesn't require synchronization after the execution of concurrent tasks.



When A terminates → B & C execute conc. when the 1st of them, say, B, terminates, D is activated. When C terminates, D is activated again

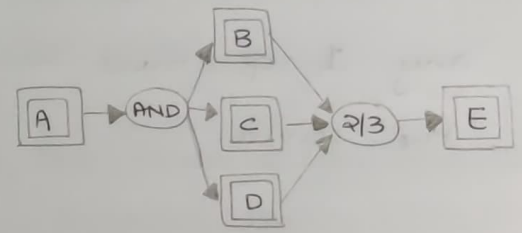
⑧ Discriminator · P

Waits for a no. of incoming branches to complete before activating the subsequent activity. Next, it resets itself.



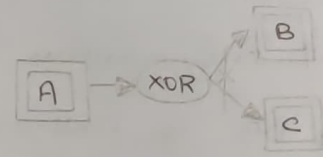
⑨ N out of M

Provides a barrier synchronization. Assuming that M>N tasks run concurrently, N of them have to reach the barrier before the next task is enabled.



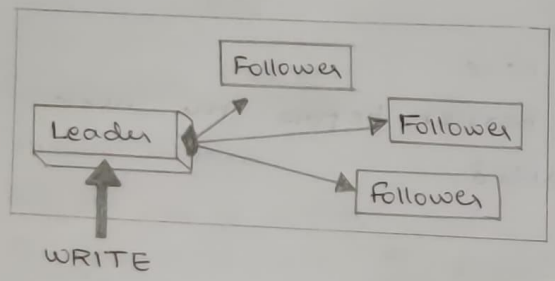
⑩ Deferred Choice · P

Similar to XOR split, but here the choice isn't made explicitly. The run-time env. decides what branch to take.

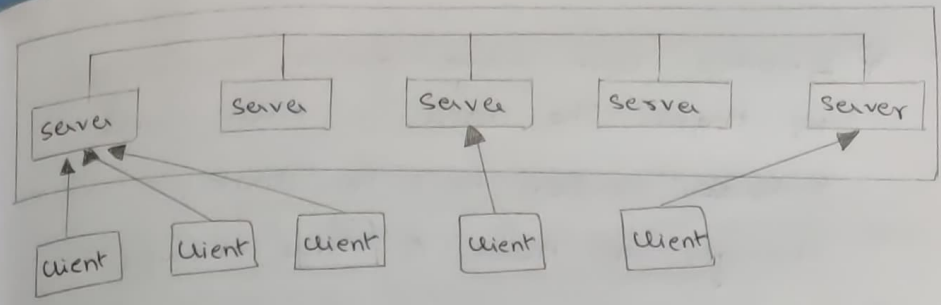


THE ZOOKEEPER

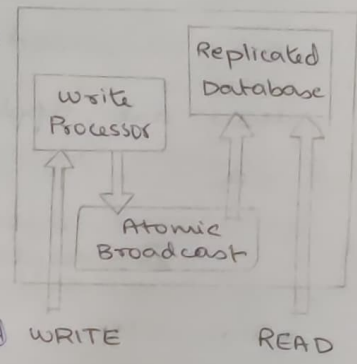
* It is a distributed coordination service
* The open-source software is written in Java & has bindings for Java & C.
* The Zookeeper software must 1st be downloaded & installed on several servers; then clients can connect to any 1 of these servers & access the service.



* Servers in the pack communicate w/ one another & elect a leader.
* A DB is replicated on each one of them & consistency of the replicas is maintained.



* The service provides a single system image. A client can connect to any server of the pack.
* A client uses TCP to connect to a single server. (Requests, Responses)
* A client synchronizes its clock w/ the server. If the server fails, all TCP connections connected to it time out & the clients detect the failure & connect to other servers.
* A read operation directed to any server in the pack returns the same result.
* Write op → Any follower receiving a request from one of the clients connected to it, forwards it to the leader.



* The leader (↑) uses atomic broadcast to reach consensus.
* When the leader fails, the servers elect

* Znodes can have data associated w/ them. The data in each node includes version no's for data, changes of ACLs, & time stamps.
* A client can set a watch on a znode & receive a notification when the znode changes.
* A read returns all the data stored in a znode, whereas a write replaces all the data in a znode
* Zookeeper data, the image of the state is stored in the server memory. Updates are logged to disk for recoverability, & writes are serialized to disk before they are applied to the in-memory DB.

Zookeeper service guarantees:

* Atomicity
* sequential consistency of updates
* Persistance of updates
* Single system image for clients.
* Reliability.

MAPREDUCE

* In case of transaction processing systems, a front-end system distributes the incoming transactions to a no. of back-end systems to balance the load.

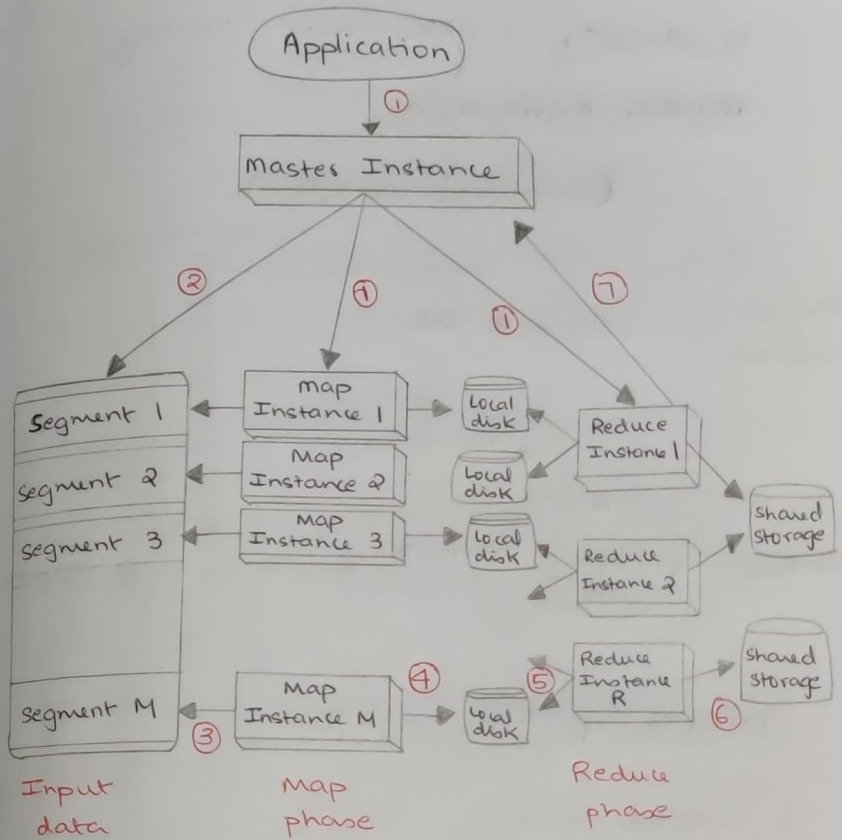
Types of divisible workloads

- Modularly divisible workload partitioning is a priority
- Arbitrarily d
  The workload can be partitioned into an arbitrarily large no. of smaller workloads.

* MapReduce is based on a very simple idea for parallel processing of data-intensive app^n's supporting arbitrarily divisible load sharing.
* As a result of computation, a set of i/p <key, value> pairs is transformed into a set of o/p <key, value> pairs.

When a user program invokes the MapReduce function, the following sequence of actions take place:

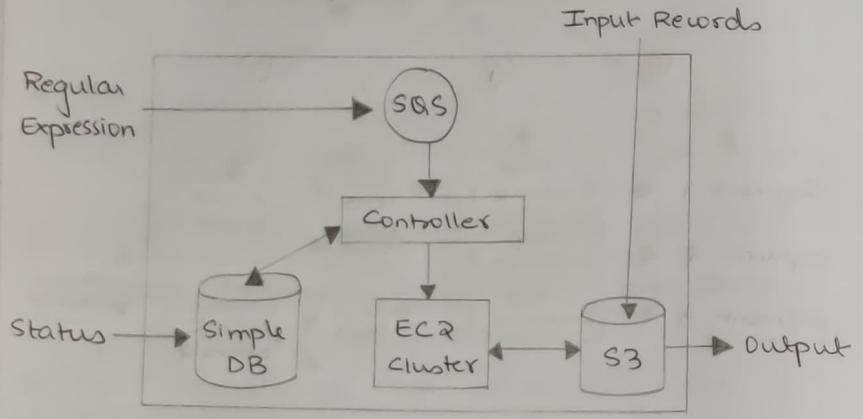
* An app^n starts a master instance & M worker instances for the Map phase, & later, R worker instances for the Reduce phase
* The master partitions the i/p data in M segments
* Each Map instance reads its i/p data segment & processes the data.
* The results of the processing are stored on local disks of the servers where the Map instances run.
* When all Map instances have finished processing their data, the R Reduce instances read the results of the 1st phase & merge the partial results.
* The final results are written by the Reduce instances to a shared storage server.
* The master instance monitors the Reduce instances &, when all of them report task completion, the app^n is terminated



GREP THE WEB

* This appⁿ performs a search of a very large set of records, attempting to identify records that satisfy a regular expression.

(simplified workflow)



Details of the workflow:

① The Startup phase

* Creates several queues → launch, monitor, billing & shutdown.
* Starts the corresponding controller threads.
* Each thread periodically polls (checks the status of) its i/p queue, & when a msg is available, retrieves the msg, parses it, & takes the required actions

② The Processing phase

* It is triggered by a StartGrep user request; then a launch msg is enqueued in the launch queue.
* The launch controller thread picks up the msg & executes the task; then it updates the status & timestamps in the Amazon Simple DB domain.
* Finally, it enqueues a msg in the monitor queue & deletes the msg from the launch queue.

Steps:

- The launch task starts Amazon EC2 instances. It uses a JRE preinstalled AMI, deploys required Hadoop libraries, & starts a Hadoop Job (MapReduce).
- Hadoop runs map tasks on EC2 slave nodes in parallel. A map task takes files from Amazon S3, runs a RE, & writes the match results locally, along w/ a description of up to 5 matches. Then the reduce task combines & sorts the results.

* Final results are stored on Amazon S3 in the o/p bucket.

③ The Monitoring phase

* The monitor controller thread retrieves the msg left at the beginning of the proc. phase, validates status/errors in Amazon Simple DB, & executes the monitor task

* It updates the status in Amazon Simple DB & enqueues msgs in the shutdown & billing queues.

* It checks for the Hadoop status periodically & updates the Simple DB w/ status/errors & the Amazon S3 o/p file.

* Finally, it deletes the msg from the monitor queue when the processing is complete

↑

It keeps checking the o/p DB to see if the o/p is complete. If it is, it triggers shutdown.

④ The shutdown phase

* The shutdown controller thread retrieves the msg from the shutdown queue & executes the shutdown task, which updates the status & time stamps in the Amazon Simple DB domain.

* Finally, it deletes the msg from the shutdown queue after processing

Steps:

- The shutdown task kills the Hadoop processes, terminates the EC2 instances after getting EC2 topology info. from Amazon Simple DB, & disposes of the infrastructure.
- The billing task gets the EC2 topology info., Simple DB usage, & S3 file & query i/p, calculates the charges, & passes the info. to the billing service.

⑤ The Cleanup phase

* Archives the Simple DB data & user info.

⑥ User interactions w/ the system

* Get the status & o/p results.
* GetStatus is applied to the service endpoint to get the status of the overall system & download the filtered results from Amazon S3 after completion