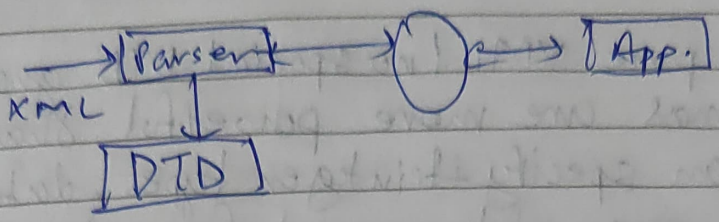
- XML Parser Processing Model (DTD)



* DTD (Document Type definition.)

- DTD defines the structure and the legal elements and attributes of an XML documen

- Two Types of Parser :-

- Validating parser -

1. Must retriere all entities and must process all DTD content.
2. Will stop processing and indicate failure.

- Non validating parser -

1. Will try to retrieve all entities defined in the DTD but will cease processing DTD content at the first entity it can't find.

- Special Rules :- Character can used as White Space in tags.
- Character Reference (&#9824) → ♠
- Binary data must be encoded as printable characters.

→ XML schemas :- (2001) →

- Use pure XML (no special DTD grammer)
- Schemas are more powerful than DTDs
  ↳ can specify it integer types, date, string
- They are often used for validation

→ XML Namespaces :-

XML namespaces provide a method to avoid element name conflict.

↳ Solving this conflict using a prefix

<h:table> </h:table> → HTML
<f:table> </f:table> → XML

- When using prefix, a namespace for the prefix must be defined.

- Namespace can be defined as xmlns attribute → in start tag

<h:table xmlns:h=" "> → HTML
<f:table xmlns:f=" "> → XML

- Also namespaces can be declared in root element.

→ XML Processing

- A) SAX: Simple API for XML

- An event based interface.
- Parse reports whenever it sees a tag/ Attribute etc.
- Event handlers are used to handle events.

* Advantages

- Simple to Use
- Very fast
- Low memory footprint.

* Disadvantages:

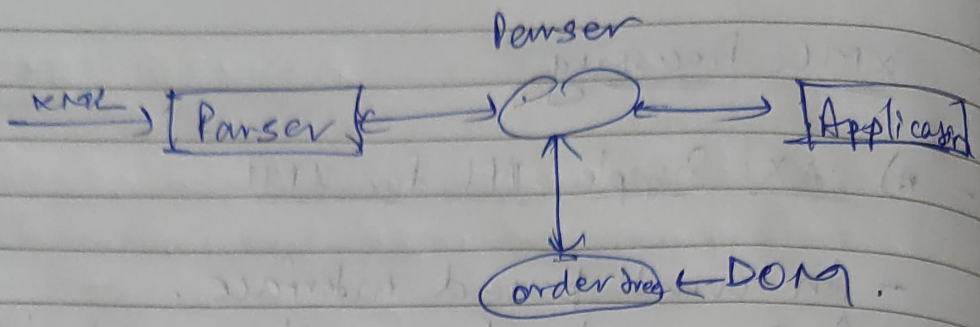
- Everything should be manage yourself.
- Not useful when you have to work dynamically.

- B) DOM (Document object Model)

- An Object based interface.
- Parser generates in memory tree.

* Advantages:- Very useful for dynamic modification
- Same interface for diff. language.

* Disadvantages:- can be slow
- DOM programming is bit akward. Not full object Oriented.



- c) JDOM :- Java document Object Model.

- Parser generates an in-memory tree for accessing & modifying the tree.
- JDOM interface has methods for accessing & modifying the tree.

* Advantages :-

- Very useful for dynamic modification of tree.
- Useful for querying.
- Nicer OOPs interface than DOM.

* Disadvantages:

- Can be slow and take up lots of memory.
- New, not entirely cooked.
- Only work with Java (not core Java).

- D) XSLT - (eXtensible Stylesheet lang.)

- An XML lang. for processing XML.
- Does tree transformation

* Advantages - Useful for tree transformation.
- Query a document.

* Disadvantages :- Can be slow for large files.
- Can be difficult to debug.

→ XML messaging :-

- Use XML as format for sending messages.
- Advantages :- Common Syntax.
- Can be use common transport mechanisms to move the XML data. (HTTP, HTTPS)

* Requirement

- Shared understanding of dialects for transport.
- Shared acceptance of messaging contract.

* Disadvantages :

- Asynchronous transport ; no guarentee of delivery.
- Messages will be much longer than binary.

→ WEB SERVICES

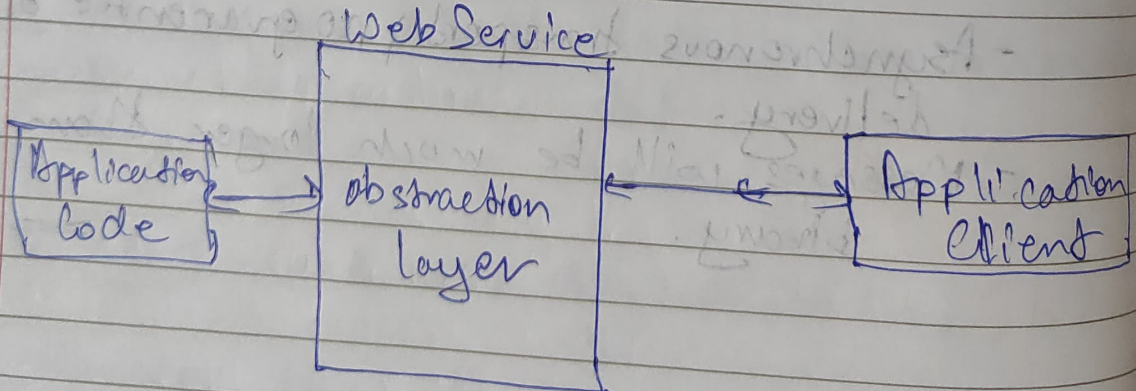
Def:- An application component that communicates via open protocols (HTTP)

(OR)

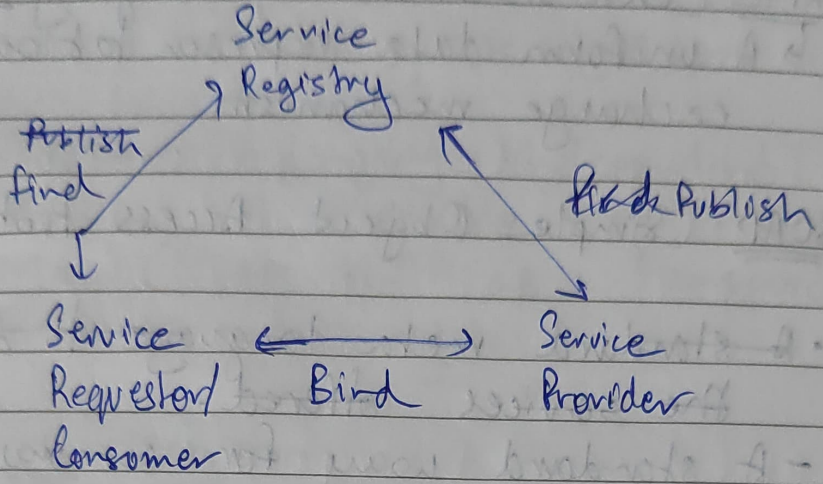
Method of communication blw two electronic devices over a network.

- Intended to solve three problems :-

1. Interoperability (Bound to Windows OS)
2. Firewall traversal → Web services uses HTTP most firewall allow access through port.
3. Complexity - Web services is developer friendly service system
   - Using different languages.



→ Web Service Model



- Service Provider :-
  - Owner of the service
  - Platform that hosts access to the service
- Service Requestor
  - Business that requires certain function to be satisfied
  - Application looking for invoking to do interaction
- Service Registry
  - Search registry of service descriptions where service provider publish their service.

- Web Service Components:-

- XML (extensible Markup language)
  ↳ A uniform data representation and exchange mechanism.

- SOAP (Simple Object Access Protocol)

  - ~~A standard meta language to describe the services offered.~~
  - A standard way for communication

- WSDL (Web Services Description language)

  - A standard meta language to describes the services offered.

- UDDI (Universal Description, Discovery & Integra

  - A mechanism to register & locate WS based application.

- Steps of Operation

1. Client queries registry to locate service
2. Registry refers client to WSDL docs.
3. Client access WSDL
4. WSDL provides data to interact
5. Client sends SOAP message request.
6. Web service return SOAP message response.

→ SOAP :-

- Simple Object Access Protocol.
- format for sending messages over internet.
- XML based
- Platform & language independent.
- Simple and extensible
- Mainly uses HTTP as a transport protocol.
- Stateless one way

* Why SOAP :-

- SOAP is platform neutral choice.
- Simple XML wire format.

* Characteristics :-

- Security and routing
- SOAP can be used over transfer protocol.
- Independent.

→ WSDL :-

- Web design Service Design language
- Contract b/w XML Web service & client.
- Defines where the service is available

- Specifies what a request message must contain

* WSDL Document Structure

→ A WSDL document is just simple XML document.

- Port type (Operation performed by web service)
- message (messages used by web service)
- types (Data type used)
- binding (communication protocol used)

→ UDDI (Universal Description, Discovery Integration)

- framework to define XML based registry.
- Register are repositories that contain document.
- Platform independent.
- Uses WSDL to describe interfaces to web services.

*Benefits:-

- Making possible to discover the right business.
- Reaching new customers & increasing access to current customers.
- Expanding offering & extending market reach.

SLA

→ SLA (Service level Agreement)

A formal contract b/w a service Provider and Service Consumer.

↳ - SLA is foundation consume trust in provider

- Purpose:

To define a formal basis for performance and availability the SP guarantee to deliver.

- SLA contains Service level Objective (SLOs)

↳ Objectively measurable condition for the service.

- SLA content

1. Set of services will be delivered.
2. Complete defn of each service.
3. Responsibilities of the provider & consumer
4. Set of metrics to measure service provided
5. Mechanism to monitor the services.
6. If terms are not satisfied what remedies are available.
7. SLA will change over time.

- Types of SLA

- Two types of SLAs

1. Off the shelf or non-negotiable SLA or Direct
2. Negotiable SLA.

*1. Non Negotiable :-

- Provider creates SLA template and define all criteria via. contract, billing etc

*2. Negotiable :-

- Negotiable via external agent..
- Negotiable via multiple external agent.

- SLO (Service level Objective)

- Checks multiple parameter

Ex:- Availability of service X is 99.9%.
- Response time of DB is 3 to 5 sec.
- Throughput of a server peak load time is 0.875.

- SLM (Service level Management)

↳ Monitoring & measuring performance of services based on SLOs

- Consideration for SLA :-

- Business objective.
- Responsibilities of Provider & Consumer
- System redundancy.
- Maintenance
- Location of data.
- Seizure of data.
- failure of the provider
- Jurisdiction.

- SLA requirements

- Security (P, I, S)
- Data encryption (P, I)
- Privacy (P, I, S)
- Data Retention and deletion (I, S)
- Hardware Erasure and Destruction. (I, S)
- Transparency (P, I, S)
- Certification (P, I, S)
- Monitoring (P, I, S)
- Auditability. (P, I, S)

- P → (Paas)
- I → (Iaas)
- S → (Saas)

Example Amazon

SLA

Service → EC2 → Iaas → Service 365 days, Credit. Availability (99.95%).

→ S3 → Storage → Availability (99.5%), as a service Error rate.

→ DB → Does not gurantee availability

→ Limitations

→ Service Measurement - Restricted to uptime percentage.
→ Biasness towards vendors
→ Lack of active monitoring on customer side.
→ No formal way of verify if SLA gurantee are complying or not.

→ ECONOMICS :-

- Cloud Properties Economic Viewpoint

1. Common Infrastructure

- Pooled, standardized resources.

2. Location Independence
- Meeting performance requirement with benefits deriving from latency reduction

3. Online Connectivity
- An enabler of other attributes ensuring service access. Cost and performance impacts of network.

4. Utility pricing
- Usage - sensitive or pay per use pricing

5. On-Demand Resources
- Scalable, elastic resources provisioned and de provisioned without delay.

- Economical backgroud of cloud

- Pay as you go model offered by cloud provider
- Scalable and simple.

- Cloud computing allows :-

- Reduces the capital cost of infrastructure.
- Removes maintenance cost.
- Removes administrative cost.

- Value of location independence :-

- Service and content comes to us Through : Wired, wireless, satellite.
- Supporting Global user requires coordination, consistency, availability.

- Value of Utility pricing :-

- Economy of scale might not be very effective.
- Cloud services don't need to be cheaper to be economical.
  It depends on demand.

- Value of on demand services :-

- Pay a penalty whenever your resources do not match the instantaneous demand.

$D(t)$ - Instant demand at time t
$R(t)$ - Resources at time t

→ Penalty Cost $\propto \int |D(t) - R(t)| \, dt$

- If, demand is fixed, penalty = 0

→ MANAGING DATA

- Cloud data management is a way to manage data across cloud platform

→ RDBMS
- Default data storage and retrieval mechanism
- Efficient in transaction processing.
  Example → System R etc.

→ For scalable web search service

- Google file System (GFS)
- Big Table (Organizes data)
- Map Redvee (Parallel programming).

→ Suitable for
- Large volume

→ Similar to BigTable data model
- Google App Engine's Datastore
- Amazon's Simple DB.

→ In Relational Databases

- User application programs interact with an RDBMS through SQL.

- RDBMS parser:
  - Transforms queries into memory
  - Optimizes execution time

- Disk space management layer
  ↳ Stores data records on pages of contiguous memory blocks.

- Database file system layer
  - Independent of OS file System

- Data Storage Techniques

1. Row-oriented storage.
   - Optimal for write-oriented operation.

2. Column-oriented storage.
   - Efficient for data-warehouse workloads.

* - Parallel Database Architecture

- Shared memory
  ↳ Suitable for servers with multiple CPU's
- Connected by a network.
- Hybrid Architecture.

* Advantages of Parallel DB over Relational DB

1- Efficient execution of SQL queries by exploiting multiple processors.
2. Tables are partitioned & distributed across
3. Distributed two phase commit locking for transaction
4. fault tolerant.

- Traditional transaction DB : Oracle, SQL
- Data warehousing DB : Netezza, Vertica.

- Cloud file System :-

- GFS (Google file System)
1. Design to manage large files
2. Handles
   - failures even during reading or writing.
   - fault tolerant
   - Supports parallel read and write.

- Hadoop Distributed file System (HDFS)
1. Open Source implementation of GFS
2. Available on EC2 cloud platform.

- Datastore :-

- Google & Amazon offer simple transactional <key, value> pair database.

- Google App Engine Datastore
- Amazon Simple DB.

- Entities table: store data as one column
- Multiple index tables are used to support queries.
- Bigtable - Horizontally partitioned across disk
  - sorted by the key - value.
- Efficient execution of prefix & range queries on key value.
- Entities are grouped for transaction purpose
- Support different queries.
- Automatically created indexes.

→ INTRODUCTION TO MAPREDUCE :-

↓

Mapreduce model developed at google:

- Objective → Implement large scale search.
  → Text processing on massive scalable

- Design for processing & generating large volumes of data.

- fault tolerant: ensure progress of computation even if processor & network fa

Example:- Hadoop: Open Source Implementation (developed at Yahoo)
- Available on pre-packaged AMIs on Amazon EC2 cloud platform

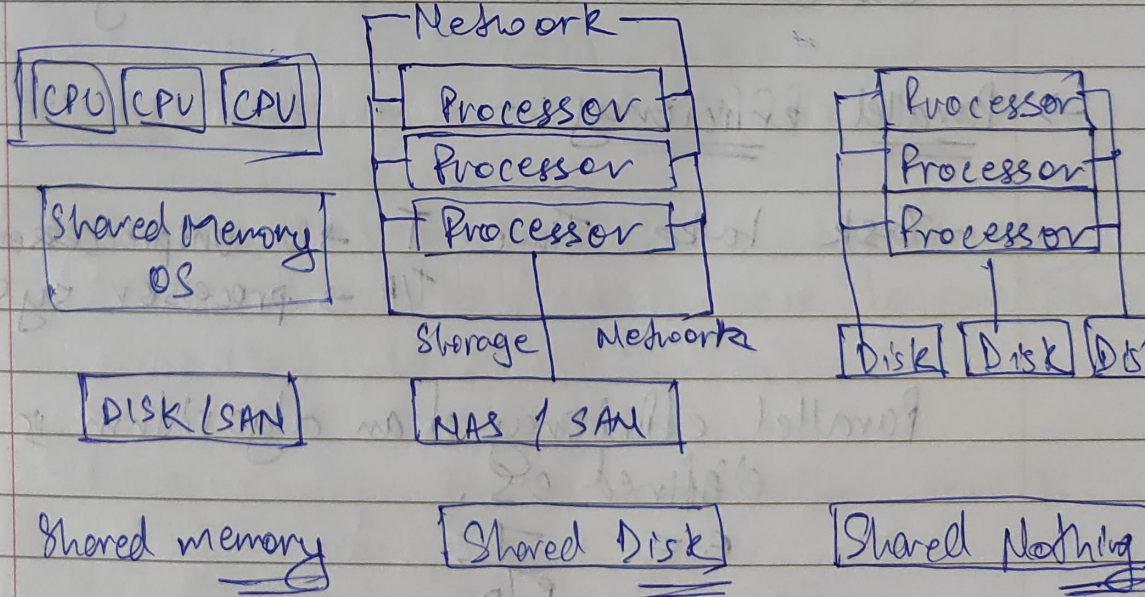
→ Parallel Computing

→ Different models of ~~pat~~ parallel programming.
  ↳ Nature & evolution of multiprocessor computer architecture.
    - Shared memory model.
    - Distributed memory model.

→ Parallel Computing:-

- Developed for compute intensive tasks.
- later found application
  ↳ shared memory
    - Shared disk
    - Shared nothing.

→ Parallel Database Architecture :-



1. Shared Memory

- Suitable for servers with multiple CPUs
- Memory address space is shared by multiple processing OS.
- Schedules processes in parallel exploting all the processor.

2. Shared Nothing

- Cluster of independent servers each with its own disk space
- Connected by a network.

3. Shared disk :-

- Hybrid architecture.
- Clusters are connected to storage.

→ Parallel Efficiency :-

- Task take Time T → in uniprocessor system
  T/P → processor system.

Parallel efficiency of an algorithm is defined as.

$$\epsilon = \frac{T}{P T_P}$$

→ MAP REDUCE MODEL :-

- Parallel programming abstraction
- Used by many different parallel application
- leverage a common underlying fault tolerant implementation.

Two phases of MapReduce :-

- Map Operation
- Reduce Operation

↳ A configurable no. of 'M' mapper & 'R' reducer processor are assigned to work on problem.

- Computation is coordinated by single master process.

• Map Phase :-

- Map operation consists of transforming one set of key value pair to another

$$Map : (k_1, v_1) \rightarrow [(k_2, v_2)]$$

- Each mapper writes computation results in one file
  ↳ Stored by local file system.
  ↳ Master keeps track of the location of these files.

• Reduce phase :-

- The masters informs the reducers where computation have been stored on local files

Reduce: $(k_2, [v_2]) \rightarrow (k_2, f(v_2))$

→ final result are written back to the GFS file system.

- Map Reduce : fault tolerance :-

1. Heartbeat communication.
   - updates are regarding the status assigned.
   - Communication exists but no progress.

2. If mapper fails the master reassigns the key-range designed to it to another working node for re execution.

3. If reducer fails, only the remaining tasks are reassigned to another node.

- Map Reduce : Efficiency.
  - Map stage : mapping time = $\epsilon_m D$
    Data Proceed as output = αD
  - Reduce stage : Reducing time = $\epsilon_r \alpha D$
    $= \alpha \epsilon_r D$

- volume of data 'D'
- 'wD' time on a uniprocessor
- 'c' time to read/write

- Consider no overheads in decomposing a task into map & reduce

$$wD = cD + cmD + crD + coqD$$

- P processor

- Time to read data from disk by mapper = $\frac{wD}{P}$

- Data produced by each mapper = $\frac{\sigma D}{P}$.

- Parallel efficiency of the Map Reduce

$$\varepsilon_{MR} = \frac{wD}{P\left(\frac{wD}{P} + 2c\frac{\sigma P}{P}\right)} = \frac{1}{1 + \frac{2c}{w}\sigma}$$

→ Map Reducing Application

1. Indexing a large collection of document
2. Relational operation using MapReduce.
   - large scale, fault tolerance.