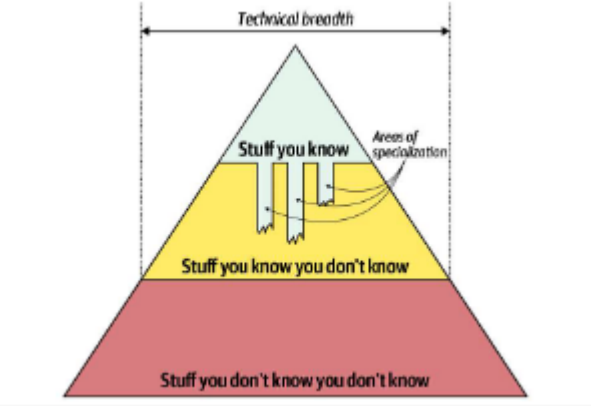
Software Architecture and Design

Chapter 1

- **Why architecture ?**
  - Large scale software system
  - Business-critical, safety-critical and mission-critical requirements
  - Large application domain
  - Meeting keys aspects of large scale software systems
- **What's software architecture ?**
  - Software Architecture is high level of abstraction and description to the structure and organization of system software.
  - Blueprint for software development and evolution.
- **What's the difference between software architecture and software design ?**
  - Software Architecture
    - Conceptual
    - Skeleton (backbone) of system
    - Common (general) attributes of a set of architectural design
    - Provide way to predict performance of the system
    - Set of principles
  - Software design
    - Plan (Instruction)
    - Blueprint of components
    - Description of structure of a system
    - Provides way to assure functionality of the system
    - Set of solutions
- **What are the major types of software architecture ?**
  - Logical architecture
    - Description of functional relationship between software components.
  - Physical architecture

        - Mapping of software component to hardware components to describe deployment process of the system.
    - Technological architecture
        - Description of SDKs and tech standards used in system development.
- **What are the 4 major aspects of architectural thinking ?**
    - Understanding the difference between Software Architecture and Software design
    - Breadth technical knowledge
        - Unlike developers technical breadth is more important than technical depth for architects because architects must make decisions that match capabilities to technical constraints, a broad understanding of wide variety of solution is valuable.



    - Understanding, analyzing tradeoffs between different technologies.
    - Understanding importance of business drivers and how they translate to architectural concerns.
- **What's modularity in software architecture ?**
    - Set of standardized parts or independent unit that's used to construct complex structure in system development.
- **What are measures of modularity ?**
    - Cohesion
        - Describes how the elements of the system belong together.
        - Can be measured using LCOM (Lack of Cohesion in Methods). The lower LCOM value indicates good structural cohesion and vice versa.

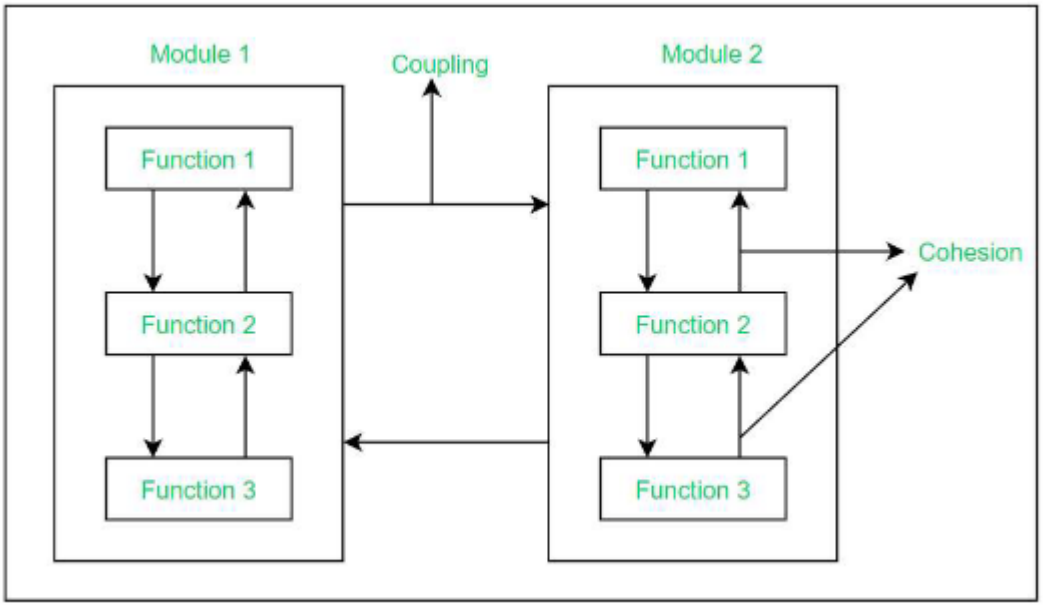
        - Functional cohesion:- every part of module is related to each other.
        - Sequential cohesion:- output of one module is input to another module.
        - Communication-al cohesion:- two modules create communication chain to work on the same output.
        - Procedural cohesion:- two modules must execute in particular order.
        - Temporal cohesion:- modules related based on timing dependencies.
    - Coupling
        - Indicates degree of interdependence between software modules.
        - Data coupling:- degree to which one module depends on data from another module. It measures the amount of data that is passed between modules.
        - Stamp coupling:- two modules are connected through a shared global data structure, such as a flag or a counter.
        - Control coupling:- the degree to which one module controls the flow of another module. It is a measure of how much one module relies on another module to make decisions or perform actions.
        - Common coupling:- the degree to which two or more modules share the same global data. This means that changes made in one module can affect other modules that use the same data.
        - Content coupling:- the degree to which one module depends on the internal workings of another module.
    - Connascence
        - Measure of a change in one module would require modification in other module in order to maintain the overall correctness of the system.
        - Static connascence:- the relationship between two or more elements in a program that can be determined at compile-time. This means that the relationship is fixed and cannot be changed during runtime. For example, if a function requires a specific input parameter type, any changes to that type will result in a compile-time error.
        - Dynamic coonascence:- the relationship between two or more elements in a program that can only be determined at runtime. This means that the relationship is not fixed and can change during runtime. For example, if a

function relies on an external resource such as a database or network connection, any changes to that resource may cause errors during runtime.

- **static connascence** is preferred over **dynamic connascence** as it allows for easier debugging and maintenance of code. However, some situations may require dynamic connascence due to the nature of the problem being solved.

- **What's the difference between cohesion and coupling ?**

Cohesion	Coupling
• Relationship between components with a module	• Relationship between 2 or more modules
• Measures module's functional strength	• Measures module's dependency with others
• it's based on communication between two elements or fields within a single module (intra-module concept)	• It's based on communication between two modules (inter-module concept)
• Possible to achieve full cohesion	• Not possible to achieve no coupling among modules



- **What are the 3 major architecture characteristics ?**
  - Operational characteristics refer to the functions and processes that a system or structure performs. This includes things like how it operates, how it is maintained, and how it interacts with other systems.
  - Structural characteristics refer to the physical components and design of a system or structure. This includes things like the materials used, the layout, and the overall shape and size.
  - Cut off architectural characteristics refer to the features of a structure that are designed to prevent or limit access. This includes things like walls, fences, gates, and other barriers that are intended to keep people or animals out.
- **How can we identify architectural characteristics ?**
  - Extracting from domain concerns
  - Requirements
  - Implicit domain knowledge
- **What are methods of measuring characteristics of architectures ?**
  - Operational measures :- of architecture refer to the performance and efficiency of the system in meeting its intended goals and objectives. This includes metrics such as response time, availability, reliability, scalability, and security.
  - Structural measures :- of architecture refer to the design and organization of the system components and their relationships. This includes metrics such as modularity, cohesion, coupling, and complexity.
  - Process measures :- of architecture refer to the methods and practices used in developing and maintaining the system. This includes metrics such as development time, cost, quality assurance, testing effectiveness, and maintenance efficiency.
- **What is governance and fitness functions in software architecture ?**
  - Governance :- in software architecture refers to the set of policies, procedures, and guidelines that are put in place to ensure that software systems are developed and maintained in a consistent and effective manner. This includes things like security protocols, compliance with industry standards, and adherence to best practices.

    - Fitness functions :- are metrics or criteria used to evaluate the performance of a software system. These functions can be used to measure things like speed, reliability, scalability, and maintainability. By using fitness functions, architects can ensure that their systems are meeting the needs of their users and stakeholders.
- **what's the scope of architectural characteristics ?**
    - The scope of architectural characteristics refers to the various attributes or qualities that define the architecture of a system or application.
- **What are key issues in software design ?**
    - Concurrency
    - Control and Handling events
    - Data persistence
    - Distribution of components
    - Error and Exception handling
    - Fault tolerance
    - Interaction and presentation
    - Security
- **Evolution of architecture**
    - No architecture stage (1946 - 1969)
    - Infant stage (1970 - 1980)
    - Junior stage (1980- 1990)
    - Advanced stage (since 1990)

Chapter 2 : Architectural styles

- **What are the various patterns for Software Architecture ?**
    - System pattern:- presents the macroscopic system pattern for software applications.
    - Architecture pattern:- architectural patterns used to design and build the software (system).

- Component pattern:- the design reuse of class (module) that carries certain type of structure and behavior.
- Data model:- abstraction of data records through data structure design that can be used for further processing and manipulation.

- **What's Software Architecture Style ?**
  - Description of common structure and a shared set of attributes for group of software architectures.
- **What are the major types of Software Architecture ?**
  - Data flow
    - Batch processing
    - Pipe-filter
  - Data centered
    - Repository design
    - Blackboard architecture
  - Hierarchical
    - Layered architecture
    - Main (Subroutine)
  - Interaction Oriented
    - MVC (Model-View-Controller)
    - PAD (Presentaion-Abstract-Control)
  - OOP
    - OOAD
    - UML Model
  - Service Oriented Architecture
  - Distributive
    - Client-Server
  - Component based
    - Framework approach

- **What's the basic concept of 3 tier client-server architecture ?**
  - Type of distributive architecture where system component is divided into UI (presentation) tier, application tier and data (database tier).
- **What are the pros and cons of client-server architecture ?**

  Pros
  - Good data distribution
  - Architectural transparency
  - Flexible deployment

  Cons
  - Limited scalability
  - High development and maintenance cost
  - Traffic congestion

- **What's the difference between client-server and browser-server architecture ?**

  Client-Server
  - Architecture where client sends requests to server and server responds with data or service.
  - Client can be any device with network access.
  - Most processing happens in server side.

  Browser-Server
  - Architecture where client uses HTTP and HTTPS protocols to fetch data from server in the form of HTML, CSS and Javascript.
  - Client is a browser.
  - Most processing happens in the client side.

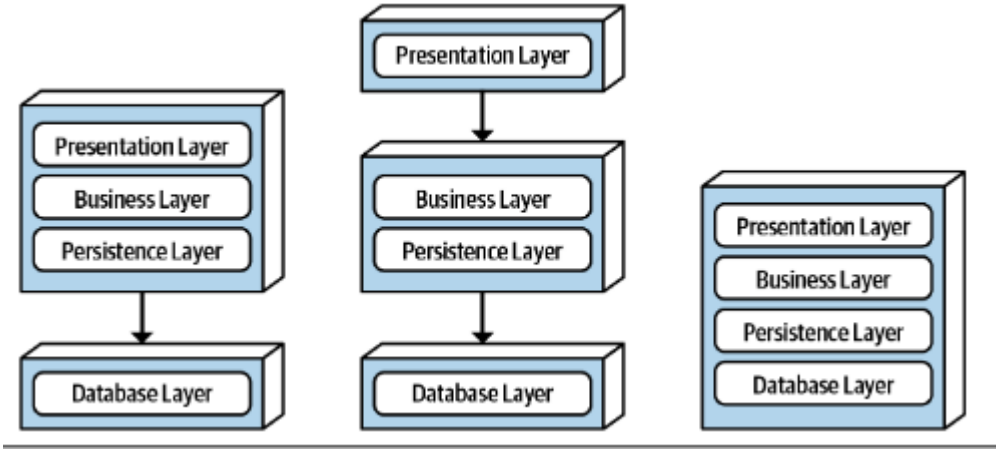
- **What are the pros and cons of browser-server architecture ?**

  Pros
  - Easy to use and maintain
  - Easy build
  - Cross platform
  - Low cost
  - Improved scalability

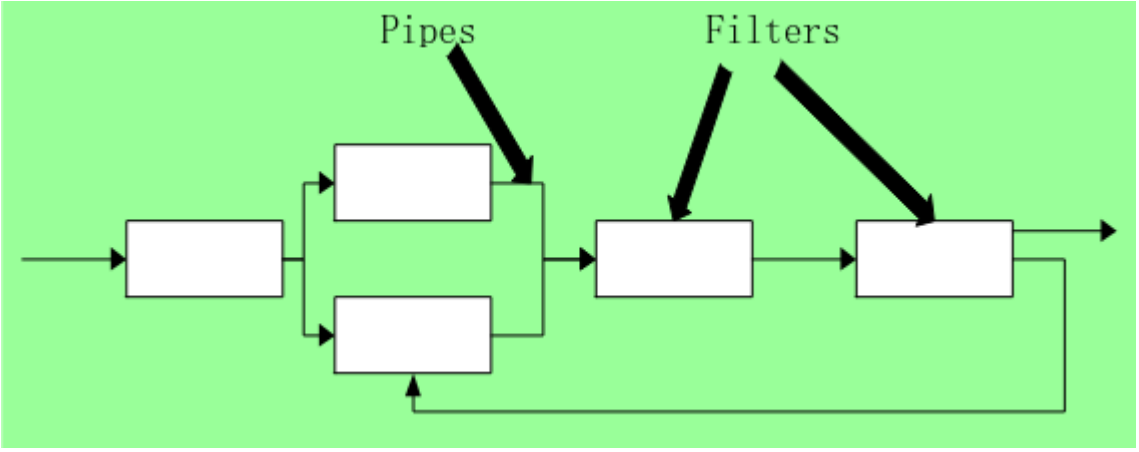
  Cons
  - Server is a single point of failure
  - Not suitable for online transaction processing system
  - Less secure

- **What is the basic concept behind layered architecture ?**
  - Architecture that divides an application into separate layers, each with a specific responsibility and level of abstraction.

- The layers are arranged in a hierarchical order, with each layer building on top of the one below it. This approach helps to improve the scalability, maintainability, and flexibility of the application.



- **What's the basic concept behind pipe-filter architecture ?**
  - Architecture where each functional unit has a group of input and output interface.



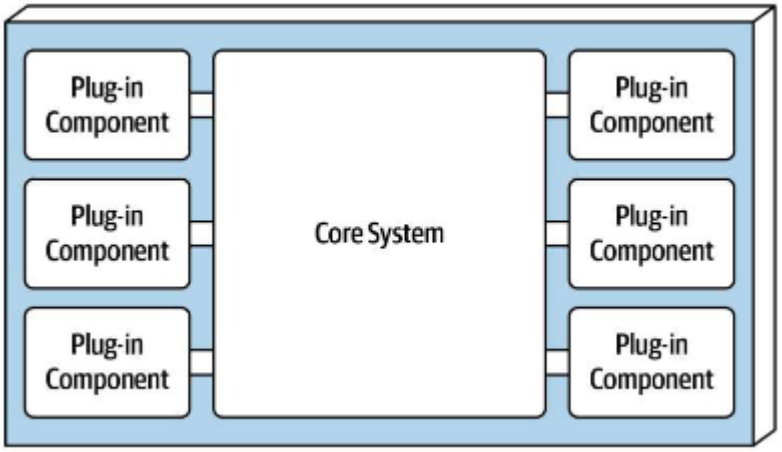
  - Filter
    - Independent functional unit that doesn't have impact on other functional unit (filter) state.
  - Pipe
    - Components of the system that sends order free and independent filter outputs to other filters.
- **What are the pros and cons of pipe-filter architecture ?**

Pros | Cons

- Reusability of functionality units
- Better maintenance and expandability
- Easy for some type of performance assesment
- Support data centeric concurrency

- Complex
- Performance issues
- Integration challenge
- Hard to debug
- Not suitable for Online Transaction Processing system that involve user interaction

- **What's the basic concept of Microkernel architecture ?**
  - Architecture where application logic is divided between independent plugin components and the basic core system.
  - Also known as plugin architecture.



- **What are the components of Microkernel architecture ?**
  - Core System
    - Defined as the minimal functionality of the system.
    - Implemented either by using layered or monolith architecture.
  - Plugins
    - Independent functional units with specific functions.
    - Communication with the core system is point to point by using either REST or messaging.
    - Implemented as shared libraries.

        - Can be runtime or compile time.
    - Plugin registry
        - Assist the core system to know about which plugin modules are available and how to use them.
        - Provides information about each plugin.
    - Plugin contracts
        - Behavior, input and output data returned from the plugin.
        - Implemented in JSON and XML.
- **What are the pros and cons of Microkernel architecture ?**

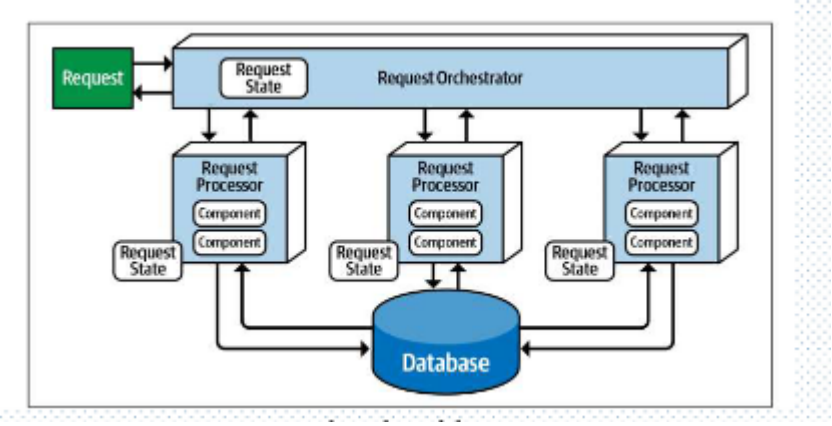
  Pros

  - Modular design
  - Flexibility
  - Security
  - Reliability

  Cons

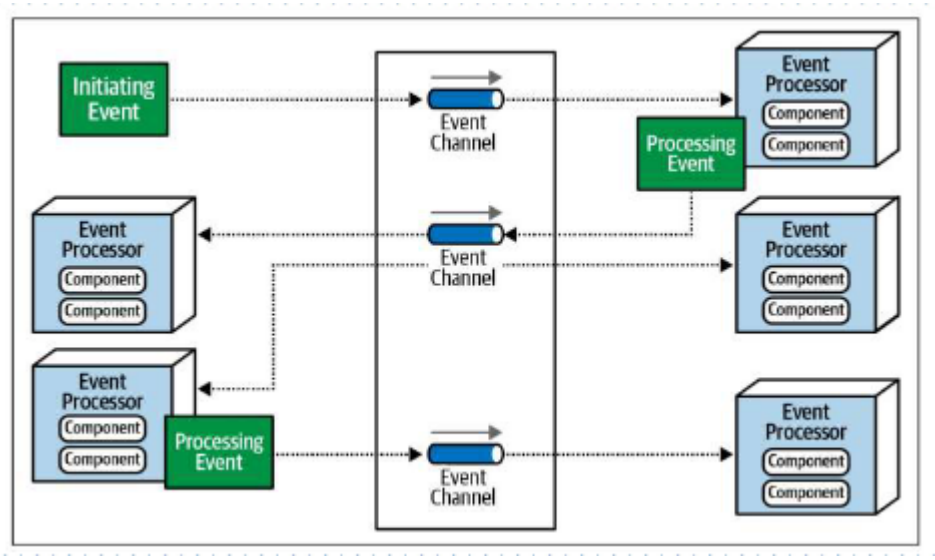
  - Performance
  - Cost
  - Complexity
  - Compatibility

- **What's basic concept behind Event Driven architecture ?**
    - Popular distributed asynchronous architecture style used ti produce highly scalable and high-performance applications.
    - Each component is autonomic.
    - Request are sent to s request orchestrator



- **What's the difference between broker and mediator topology ?**

- Broker topology
  - Commonly used when a high degree of responsiveness and dynamic control over the processing of an event is required.
  - No central event mediator.



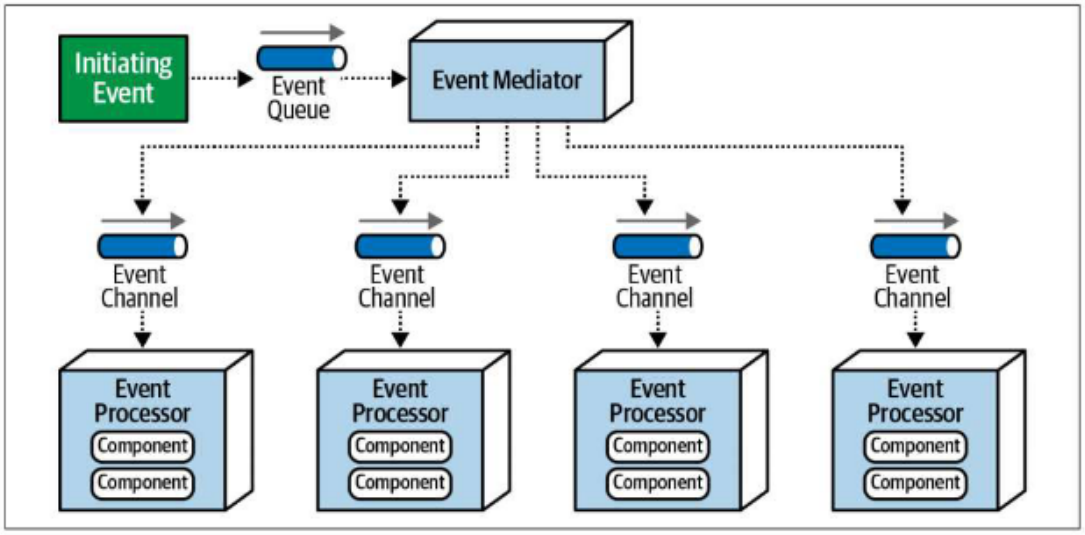
Pros

- High decoupled event processor
- High scalability
- High responsiveness
- High performance
- High fault tolerance

Cons

- Workflow control
- Error handling
- Recoverability
- Restart capabilities
- Data inconsistency

- Mediator topology
  - Commonly used when control over the workflow of an event process is required.



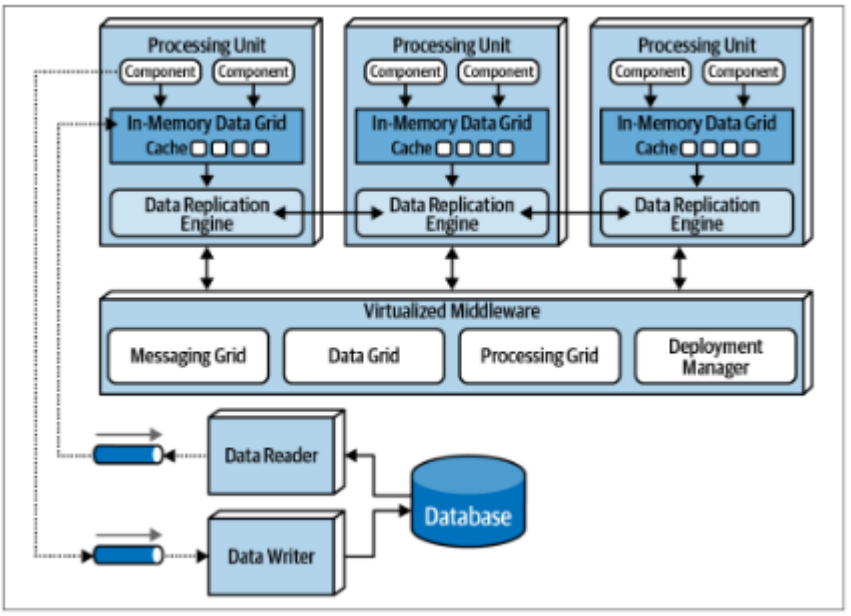
Pros

- Workflow control
- Error handling
- Recoverability
- Restart capabilities
- Data inconsistency

Cons

- More coupling of event processor
- Low scalability
- Low responsiveness (complex)
- Low performance
- Low fault tolerance

- **What's basic concept behind Space based ?**
  - Type of architecture designed to address high scalability, elasticity and concurrency issues.
  - Used for applications that have variable and unpredictable concurrent user volumes.
  - Uses multiple parallel processors communicating through shared memory.



- **What are the components of space based architecture ?**
  - Processing unit
    - Contains application logic (web based + backend)
  - Virtualized middleware
    - Handles infrastructure concerns
      - Message grid:- manage input request and state session.
      - Data grid:- manage data across processing caches
      - Processing grid:- manages orchestrated request processing.
      - Deployment management:- manages dynamic starting and shutdown of processing unit instances.
  - Data dumps
    - Data reader:- read data from database and send it to processing unit.
    - Data writer:- accepts data from data pumps and updates the database.
- **What's the basic concept behind Orchestration Driven Service Oriented Architecture ?**
  - Architecture style that delivers functionality as a service that can be used or reused when building applications.
- **What are services and service characteristics in Orchestration Driven Service Oriented Architecture ?**

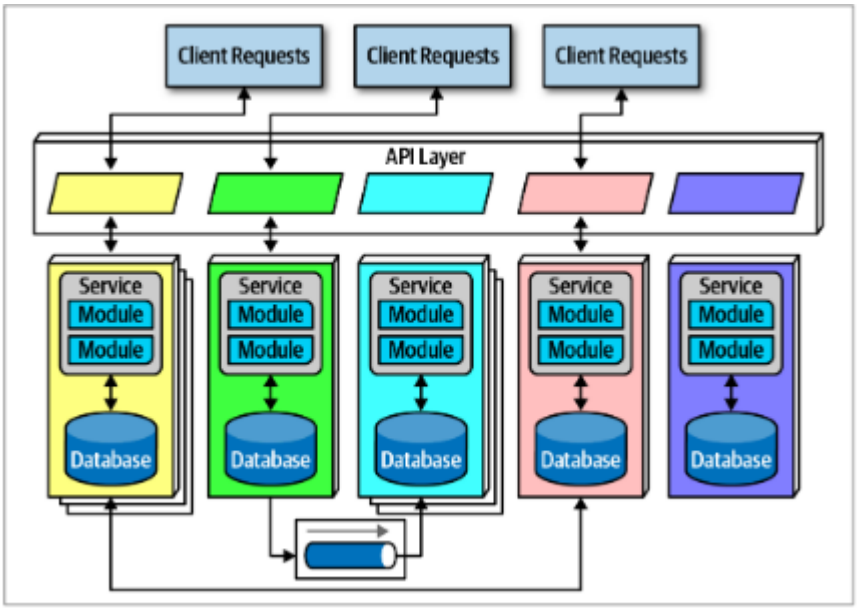
- Service is a reusable component that can be used or reused in system development.
- Service provide discrete business function that operates in data.
- Characteristics
  - **Loose coupling**:- provides well defined interfaces for clients.
  - **Stateless**:- services doesn't maintain state between invocations.
  - **Location agnostic**:- users of the service don't need to worry about the implementation details.
  - Supports open standards for integrations

- **What are the different type of service available in Orchestration Driven Service Oriented Architecture ?**
  - Business service :- top level service that provides entry point.
  - Enterprise service :- Fine grained implementations.
  - Application service :- single implementation services.
  - Infrastructure :- service that supply the operational concerns such as authentication and authorization.
- **What's Orchestration engine ?**
  - Component of Orchestration Driven Service Oriented Architecture that defines relationship between services, how they map together and where the transaction boundaries lie.
  - Acts as integration hub, allowing architects to integrate custom code with package and legacy software systems.
  - Handles message flow.
- **What are the pros and cons of Orchestration Driven Service Oriented Architecture ?**

  Pros
  - High re-usability
  - High scalability and flexibility

  Cons
  - Change in reusable components ripple out other services
  - Some services might store data not needed by other services

- **What's the basic concept behind Micro-service architecture ?**
  - Architecture style that splits logic and data access layer into several small parts called micro-services.



- **What are micro-services ?**
  - Smallest atomic unit of service that delivers business value
  - Fast, safe, independently deployed and efficiently scalable
  - Unlike monolithic micro-services doesn't share classes and components each services includes everything necessary to operate
  - Avoids coupling
  - Represent domain or subdomain in the system

Chapter 3: Cloud Computing Arcitecture

- **What's cloud computing ?**
  - Model that enables ubiquitous, convinent, on demand network access to a shared pool of configurable resources that can be rapidly provisoned (*NIST*).
  - Kind of internet based computing that provides shared processing resources and data to computers and other devices on demand (*Wiki*).

- **What are the 5 basic characteristics of cloud computing ?**
  - On demand services
  - Abstract computing resource pool
  - Pay-on-use
  - Fast and elastic expansion
  - A wide network access

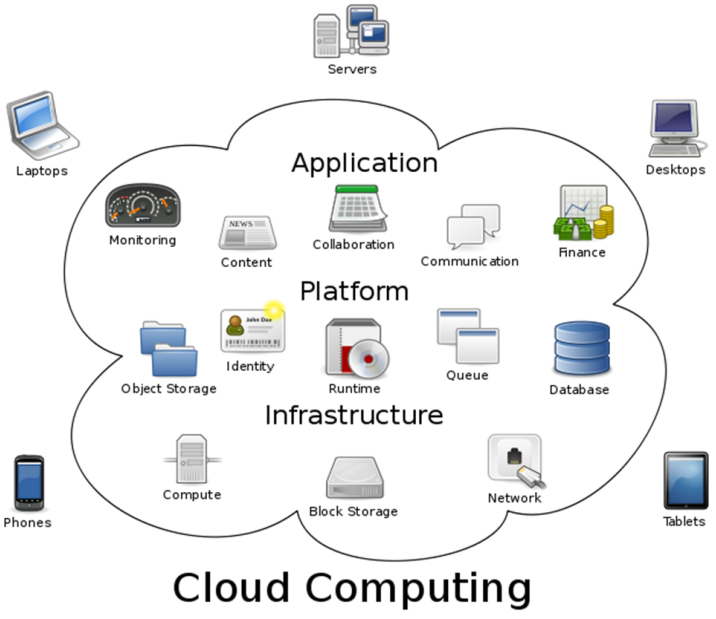
- **What are the 4 deployemnt patterns in cloud computing ?**
  - Public cloud:- cloud computing that’s delivered via internet and shared across organizations.
    - Amazon Web Services (AWS)
    - Microsoft Azure
    - Google Cloud Platform
    - IBM Cloud
  - Private cloud:- cloud computing that’s dedicated sloley to specific organization.
    - VMware
    - OpenStack
    - Nutanix
  - Hybrid cloud:- Any environment that uses both public and private cloud.
    - Microsoft Azure
    - IBM Cloud
    - Google Anthos
  - Community cloud:- shared cloud computing service environment that’s targeted to a limited set of orgnizations.
    - GovCloud: A community cloud for government agencies in the United States.
    - Healthcare Community Cloud: A community cloud for healthcare organizations that need to comply with regulatory requirements.

- **What are the 3 service models in cloud computing ?**
  - SaaS (Software as a Service):- is a cloud computing model where a third-party provider hosts and delivers software applications over the internet. The provider manages the software and infrastructure, and users access the software through a web browser or mobile application.
    - Google Workspace
    - Microsoft Office 365
    - Salesforce.
  - PaaS (Platform as a Service):- is a cloud computing model in which a third-party provider delivers a platform for users to develop, run, and manage applications. The provider manages the infrastructure and middleware, and users are responsible for the application development.
    - AWS Elastic Beanstalk
    - Google App Engine
    - Microsoft Azure App Service.
  - IaaS (Infrastructure as a Service):- is a cloud computing model where a third-party provider delivers virtualized computing resources over the internet. These resources can include servers, storage, and networking, which users can access and configure as needed. Users are responsible for managing the operating systems, applications, and data.
    - AWS EC2
    - Google Compute Engine
    - Microsoft Azure Virtual Machines.

- **What are the key elements of cloud computing ?**
  - Distributed file system and data storage architecture
  - Abstraction of computing resources and scheduling
  - Layered software architecture

- Decoupling of service interface and function implementation
- Virtual reality technology

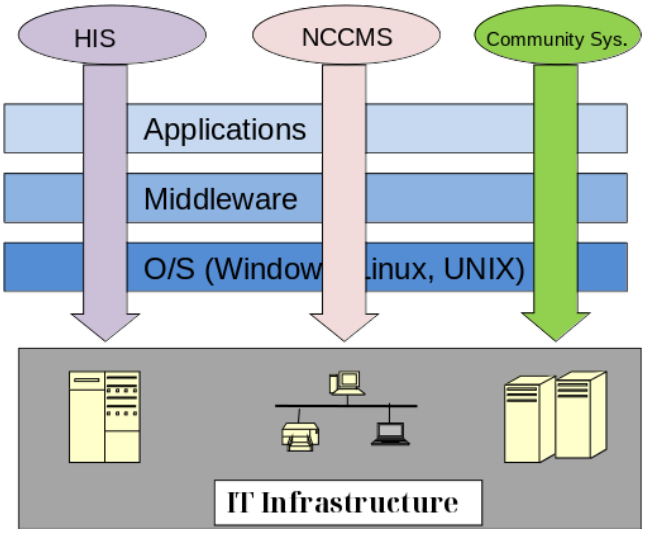
- **What's the basic concept behind layered model of cloud architecture ?**
  - Cloud applications :- various user applications that run in the cloud and serve the users.
  - Cloud platform :- the functional or service platforms that support the user application.
  - Cloud management :- system tools to schedule tasks and manage resources in cloud.
  - Cloud storage :- distributed file system and database storage to manage the data.
  - Cloud resources :- the abstraction of physical resources in the cloud.



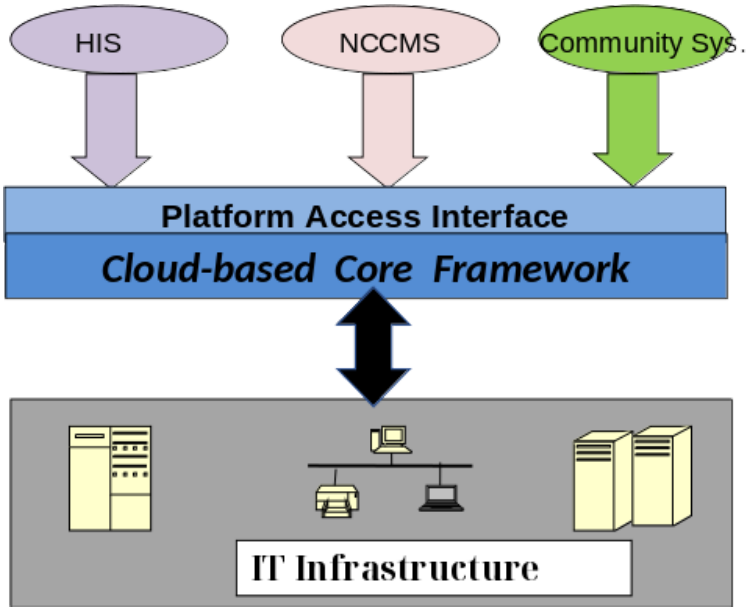
- **What are the advantages of cloud computing ?**
  - High avaliability
  - High Scalability
  - On-demand service
  - Economic

- High visibility via visualization

- **What’s isolated island cloud architectural style ?**
  - The isolated island architectural style in cloud computing refers to an approach where different applications or services are deployed in isolation from each other, similar to separate islands in the ocean. This approach is sometimes also referred to as the "siloed" or "monolithic" approach.
  - In the isolated island architectural style, each application or service is typically deployed on its own set of servers and infrastructure, and they are not directly connected or integrated with each other. This can make it easier to isolate and manage different applications or services, but it can also lead to inefficiencies and difficulties in communication.



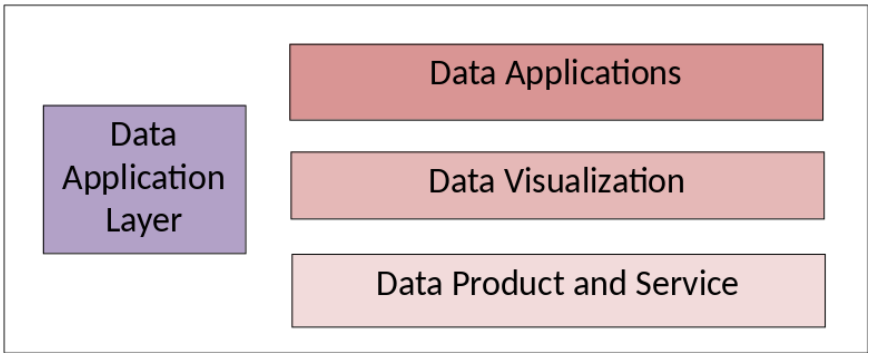
- **What’s cloud based RHIN architecture ?**
  - The cloud-based RHIN (Regional Health Information Network) architecture is a model for building a distributed healthcare information exchange infrastructure using cloud computing technologies.



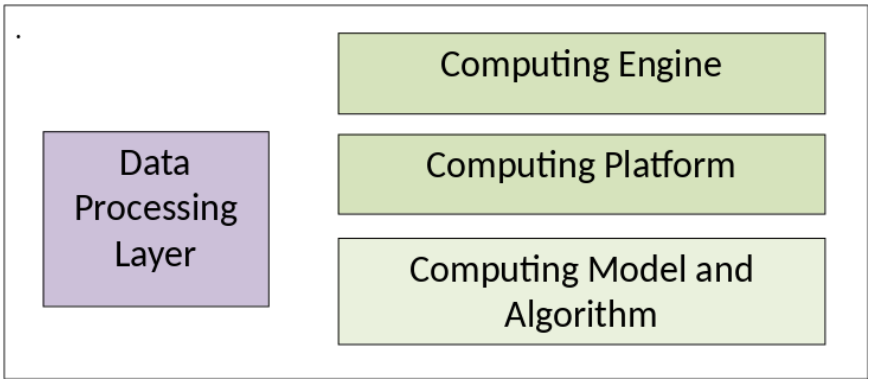
- Upper applications are independent to the bottom infrastructure which makes upgrade easier.
- Computing resources can be managed and scheduled by the platform.
- Cost effective.

Chapter 4: Big Data Computing Architecture

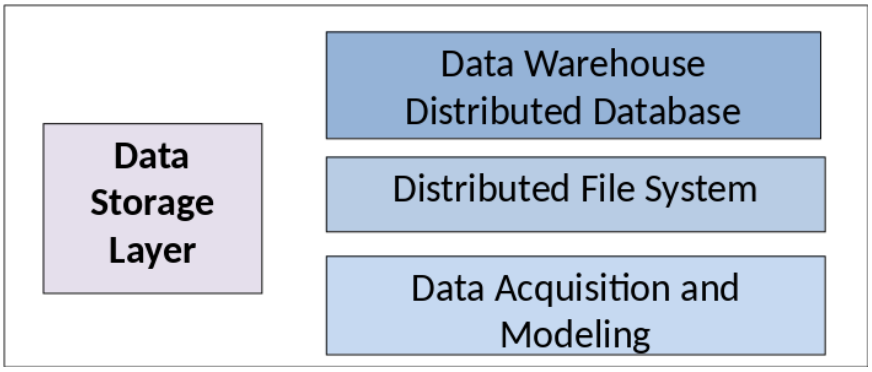
- **What are the components of big data computing layered architecture ?**
  - Data application layer: all kinds of user oriented applications and technical based on big data computing architecture.



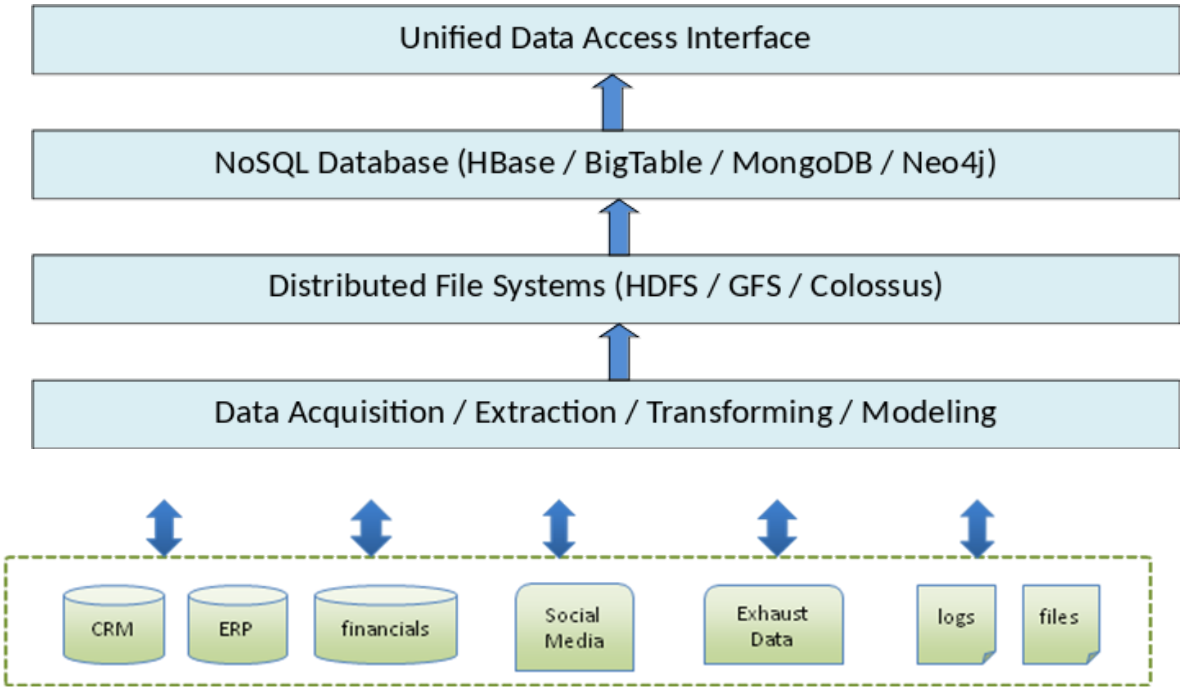
  - Data processing layer: computing models like batch processing, stream computing, analytic algorithms and computing platforms.



- Data storage layer: data storing mechanisms used in big data computing.



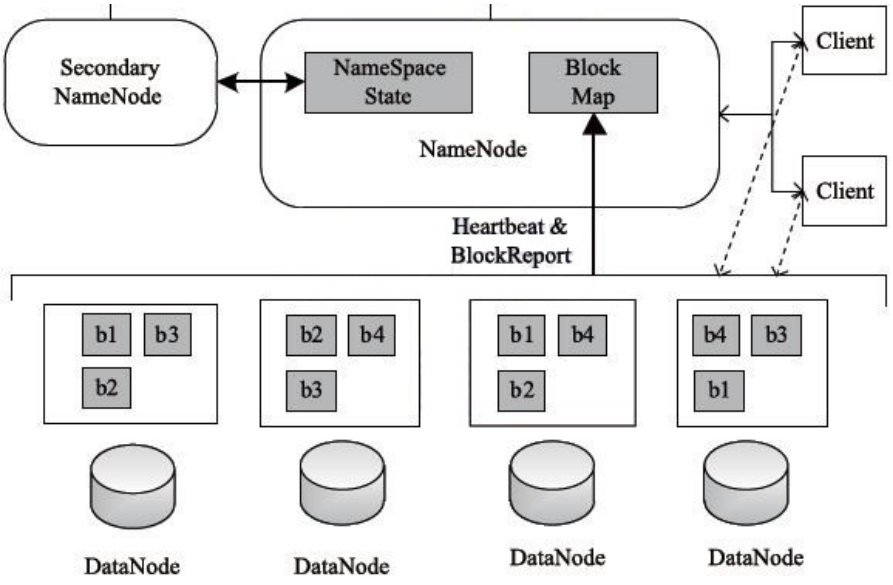
- **What are the components of the data storage layer ?**



- **What is Distibuted File System (DFS) ?**
  - A distributed file system (DFS) is a file system that allows files to be stored and accessed
    across multiple servers in a distributed computing environment.

- **What's Hadoop and HDFS ?**
  - Hadoop is an open-source software framework used for distributed storage and processing of large datasets on clusters of commodity hardware. It provides a way to store, process, and analyze large volumes of structured and unstructured data in a cost-effective and scalable manner.
  - HDFS (Hadoop Distributed File System) is a distributed file system that provides scalable and fault-tolerant storage for large datasets in a Hadoop cluster. It is designed to run on low-cost commodity hardware, making it a cost-effective solution for storing and processing large volumes of data.

- **What's Master - slave architecture ?**
  - The master-slave architecture is a key design feature of the Hadoop Distributed File System (HDFS). In this architecture, there is a single NameNode that acts as the master node and manages the file system metadata, while multiple DataNodes act as the slave nodes and store the actual data blocks.

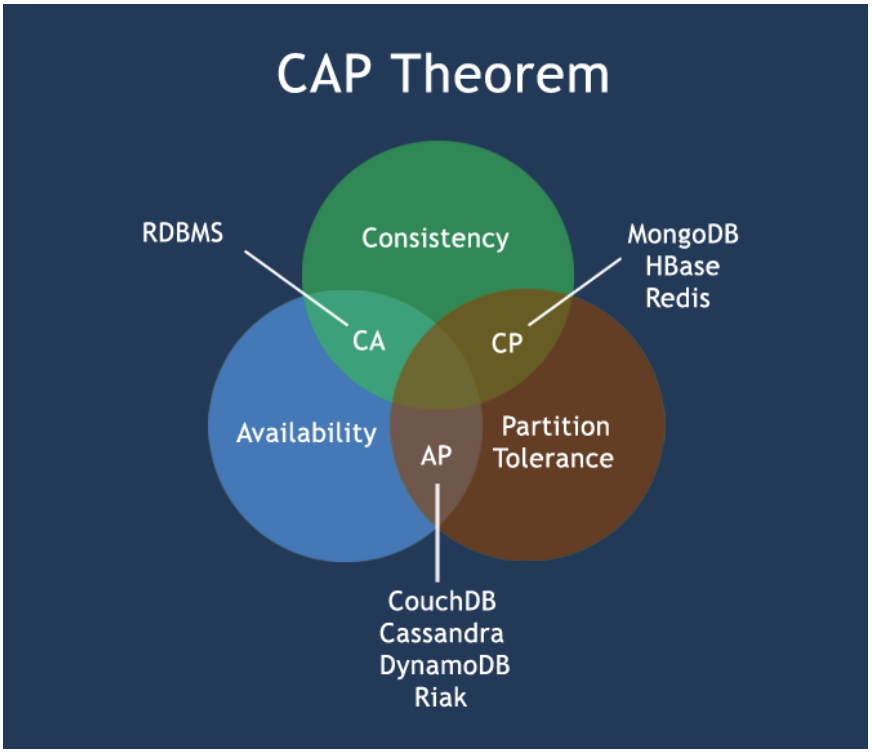


- The NameNode (Master Node) is responsible for maintaining the file system namespace, tracking the location of data blocks in the cluster, and managing access to the data by clients. It stores this information in memory and on disk.
- The DataNodes are responsible for storing the actual data blocks and responding to read and write requests from clients. Each DataNode stores a subset of the data blocks in the cluster and periodically sends a heartbeat message to the NameNode to report its status.

- **What are the pros and cons of HDFS ?**
  - Pros
    - Open source
    - Easy to develope
    - Use cheap equipment
    - Cluster recourse management
    - Good scalability
  - Cons
    - Debug time
    - Not efficient for small size files
    - Not good for single data entry updating

- **What are No SQL Databases ?**
  - NoSQL databases are non-relational databases that are designed to handle large volumes of unstructured or semi-structured data.

- **What's CAP theory ?**



- It states that in distributed system, the reading or writing operation can only meet two of the above 3 requirements, but can't meet all 3.

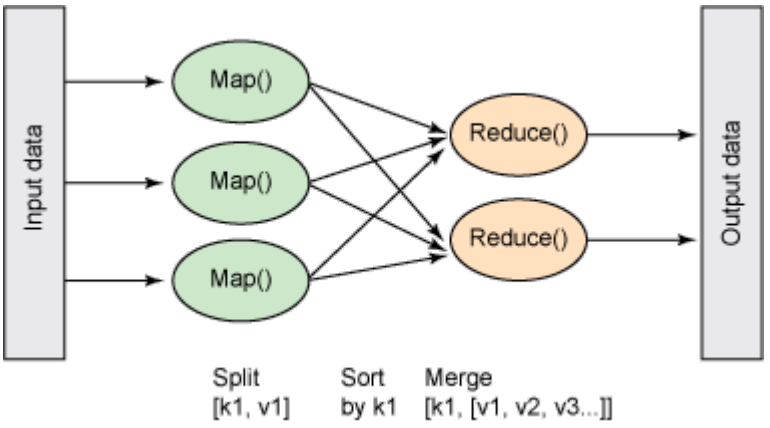
- **What are the pros and cons of No SQL databases ?**
  - Pros
    - Pre-defined data structure isn't required
    - Elasticity and scalability
    - Data partition
    - Unsynchronization for data writing
  - Cons
    - Limited query capabilities
    - Lack of standardization
    - Limited transaction

- **What are the 4 major type of No SQL databases ?**
  - Key-value
  - Column family oriented
  - Document oriented

- Graph oriented

- **What are the major computing models used in big data data processing systems ?**
  - MapReduce batch processing
  - Graph parallel computing
  - Interactive processing
  - Storm processing
  - In memory computing

- **What's the concept behind MapReduce batch processing ?**
  - MapReduce is a programming model and software framework used for processing large datasets in parallel across a cluster of computers. The concept behind MapReduce batch processing is to divide a large dataset into smaller chunks, process each chunk independently in parallel across multiple nodes, and then combine the results into a final output.



Chapter 5: Data and Transaction Design

- **What's logical data architecture ?**
  - Description of the structure of entities, relationships and constraints used in the software.

- **What are data models and application of data models ?**
  - Data model is a representation of the data structures and relationships between them that are used to store and manipulate data in a software system.
  - Data models can be used to:
    - Describe relationship between data schema and their mapping to persistent storage.
    - Define individual messages and their relationships.

- **What are the 2 types of data models ?**
  - ER model: for relational databases.
  - Object model: for non-relational (no SQL) databases.

- **What are the stable nature of data model ?**
  - Improve architecture team to participate in development of the data architecture.
  - Facilitate the rapid and early develoment of custom reports.
  - Makes a good basis for integration of components.
  - Allows the addition of new disparate functions using new technologies without impact to existing systems.

- **What’s logical data viewpoint ?**
  - A logical data viewpoint is a high-level representation of the data requirements of a system from a user's perspective. It defines the entities, attributes, and relationships between them that are relevant to the user's needs, without specifying the details of how the data is physically stored in a database.
  - A logical data viewpoint is often used to communicate the data requirements of a system to stakeholders and to ensure that the system meets the needs of its users.It

can be used to define the data requirements of a system at a high level, and to identify the key entities and relationships that are needed to support the system's functionality.

- **What are the other consideration in data model design ?**
  - Interaction with layers
    - In Model-View-Controller architecture the data model is usually a large part of the model layer.A change in the model layer is most expensive, potentially requiring changes to the upper layers of the architecture (View and Controller) to match with the data change. Incontrast change in View and Controller layer has less impact.
  - Reflection
    - Technique that allows systems to provide flexible data models that interrogate the schema at runtime and adapt accordingly.
    - Reflection facilitates the implementation of infrastructure software that provides mechanisms for marshalling , parsing and serialization of data.
    - Improve adaptability of the system.
  - Mapping objects to relational database
    - Difference in identifiers
    - Relationship between tables and objects

- **What's transaction and ACID ?**
  - A unit of interaction with persistance mechanism.
  - Atomicity: a transaction is treated as a single, indivisible unit of work.
  - Consistency: a transaction brings the database from one valid state to another.
  - Isolation: concurrent transactions do not interfere with each other.
  - Durability: once a transaction is committed, it is permanently stored in the database.

- **Why is designing transactions is important ?**
  - Poorly designed transactions can result incomplete database state.
  - Transactions span across subsystem, layer, component and even process boundaries making management of transactional data an implicit requirement for many operations to function properly.

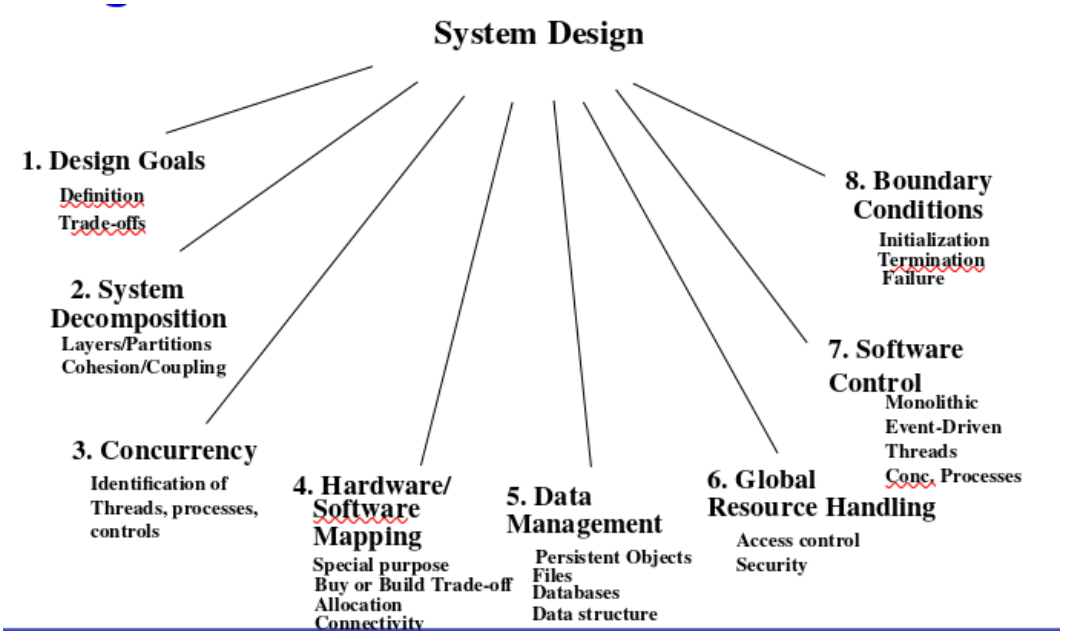
- **What are the issues to be considered when dealing with transactions ?**
  - Which component(s) control the transactions ?
  - How do transactions relate to interfaces ?
  - Where are the potential concurrency hotspots ?
  - What are the transaction rates for updates ?

- **What are the steps taken in transaction design ?**
  - Understand where transactions fit into dynamic model of the system.
  - To model the dynamics there are at least two approaches, one for analysis and one for component interactions.
  - The Component Interaction View is the component perspective on the transaction,where the Analysis Interaction View is the user perspective on the transaction.

Chapter 6: Subsystem design

- **Why system design so difficult ?**
  - Analysis: focuses on the application domain.
  - Design: focuses on the solution domain.
  - Design window: time in which design decisions have to be made.
  - Technique: time-boxed prototyping.

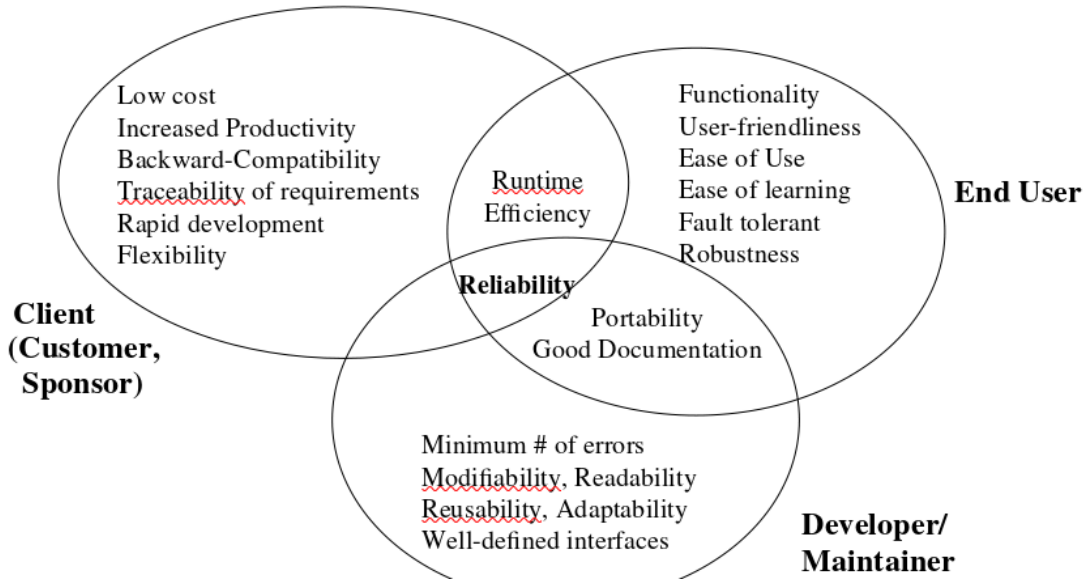
- **What are the steps in system design ?**



- **What are the requirement analysis done for system design ?**
  - Nonfunctional requirements → Design goals (step 1)
  - Functional model → System decomposition (step 2)
  - Object model → Hardware/Software mapping and Persistent data management (step 4 and 5)
  - Dynamic model → Concurrency, Global resource handling and Software control (step 3, 6, 7)
  - Subsystem decomposition → Boundary conditions (step 8)

- **What are design goals in system design ?**
  - Design goals refer to the specific objectives and requirements that a system must meet in order to be considered successful.

- **What's the relationship between design goals ?**

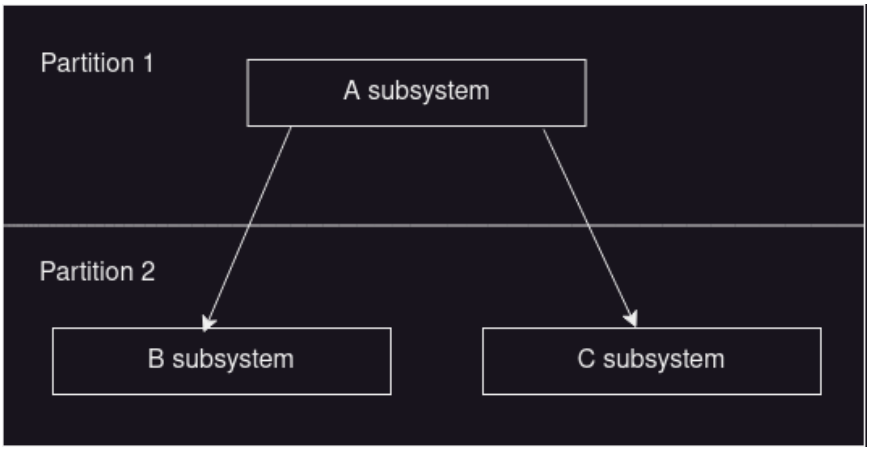


- **What are the typical design trade-offs in system design ?**
  - Functionality vs Usability
  - Cost vs Robustness
  - Efficiency vs Portability
  - Rapid development vs Functionality
  - Cost vs Reusability
  - Backward Compatibility vs Readability

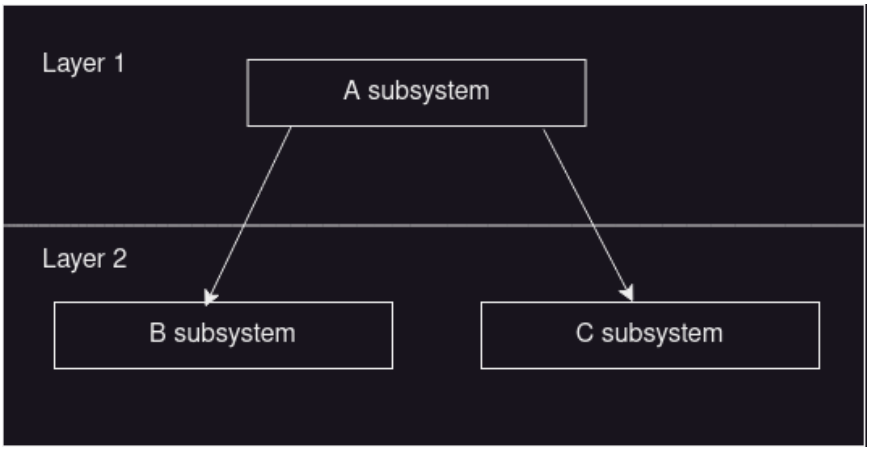
- **What's subsystem, service and subsystem interface ?**
  - Subsystem: colelction of classes, associations, operations, events and constraints that are inerrelated.
  - Services: group of operations provided by the subsystem.
  - Subsystem interface: specifies interaction and information from/to subsystem boundaries but not inside subsystem. Also known as Application Programming Interface in implementation stage.

- **What's the difference between layers and partitions in system design ?**
  - Both Partitons and Layers are techniques to achieve decupling and cohesion.

- Partitions and Partitioning
  - Collection of subsystem that perform specific functions and communicates with other subsystems to achieve overall functionality.
  - Partitioning refers to vertically dividing a large system into smaller subsystems based on functional or logical boundaries.
  - Improves system performance and maintainability.



  - As shown in the picture above subsystem A depends on subsystem B and C but subsystem B and C has no knowledge about A.
- Layers
  - A subsystem that provides a subsystem service to higher layers.
  - Also known as Virtual Machines (VM).
  - Layering refers to organizing a system into distinct levels or tiers based on the flow of data or control.
  - Improve scalability, felxibility and maintainability.

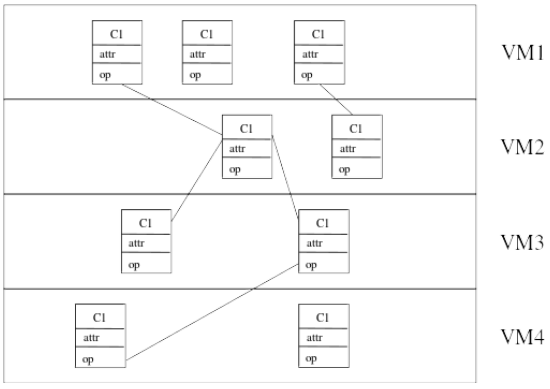


- As shown in the picture above A subsystem calls both B and C subsystem and both B and C subsystem calls subsystem A. Subsystems have mutual but not deep knowledge about each other.

- **What's the difference between open and closed layered architecture in system design ?**

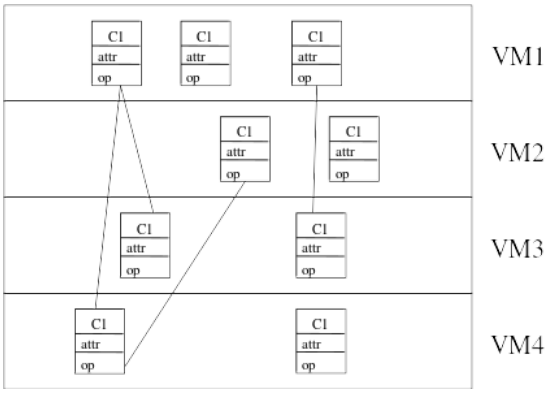
  Closed Architecture

  - Any layer can only invoke operations from immediate layer below.
  - High maintainability and flexibility.
  - Portable

  Open Architecture

  - Any layer can only invoke operations from any layer below.
  - Runtime efficiency
  - More efficient

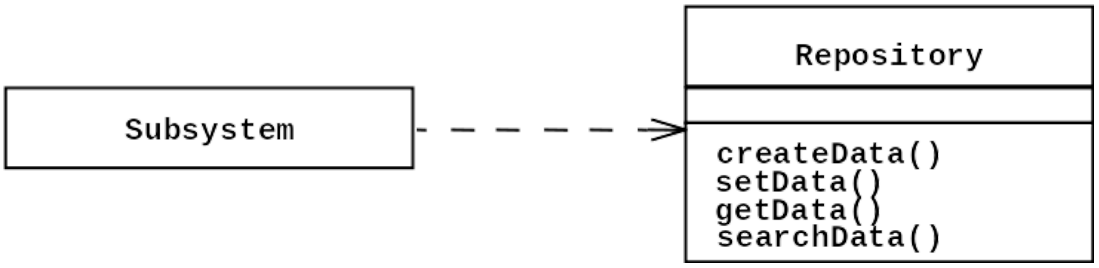
  

- **What are the properties of layered architecture ?**
  - Hirerarchical
  - Low coupling and complexity
  - Scalability
  - Reusability

- **What are the major architectural styles used in subsystem design ?**
  - Client-Server architectural style

        - One or more servers provides services to instances of subsystems called clients.
        - Design goals
            - Portability: servers can be installed and configured on a variety of machines and operating systems.
            - Transparency: servers might be distributed but should provide a single logical service to the user.
            - Performance: clients should be customized for interactive display intensive tasks and server should provide CPU-intensive operations.
            - Scalability: server should have spare capacity to handle larger number of clients.
            - Flexibility: the system should be usable for a variety of user interfaces and end devices.
            - Reliability: server should survive node or communication link problems.
    - Peer to peer architectural style (P2P)
        - Type of architectural style where clients are servers and servers are clients by sharing a portion of their own resources such as processing power, disk storage, network bandwidth and printing facilities.
        - Example: OSI model in networking
    - Model-View-Controller architectural style (MVC)
        - Model: responsible for application domain knowledge.
        - View: displays application domain knowledge to the user.
        - Controller: responsible for sequence of instructions with the user and and notifying views of changes in the model.
        - Special case of repository architecture.
        - Control flow dictated by controller.
    - Repository (Black board) architectural style
        - Subsystems access data from a single data structure.
        - Subsystems are loosely coupled.

- Control flow dictated by centeral repository.
- Example: modern compilers



- **What's the relationship between non functional requirements and design patterns ?**

Non functional requirement	Pattern
"device independent", "must support family of products"	Abstract factory pattern
"must interface with existing object"	Adapter pattern
"must deal with existing and upcoming system interfaces"	Bridge pattern
"complex structure"	Composite pattern
"must interface to an set of existing objects"	Façade Pattern
"must be location transparent"	Proxy Pattern
"must be extensible", "must be scalable"	Observer Pattern
"must provide a policy independent from the mechanism"	Strategy Pattern