CC Answers

1. HDFS to distinguish it from other generic distributed file systems.

- **Fault Tolerance**: HDFS is designed with fault tolerance as a core feature. It expects hardware failures and addresses them by:
    - **Block Replication**: Automatically replicating data blocks across multiple nodes.
    - **Replica Placement**: Strategically placing replicas in different locations to balance reliability and communication costs.
    - **Heartbeat and Blockreport**: Using these messages to monitor the health and status of DataNodes.

- **High-Throughput Access**: HDFS optimizes for high-throughput access to large data sets by:
    - **Large Block Size**: Utilizing large block sizes (e.g., 64 MB) to reduce metadata storage and enhance streaming reads.
    - **Batch Processing Focus**: Prioritizing data throughput over latency, suitable for batch processing applications with large data sets.

2. Explain the HDFS read and write operations.

Reading a File:

- A user sends an “open” request to the NameNode to get the location of file blocks.
- The NameNode returns addresses of DataNodes containing the file blocks.
- The user connects to the closest DataNode to read the first block, then repeats for all blocks.

Writing to a File:

- A user sends a “create” request to the NameNode to create a new file.
- If the file doesn’t exist, the NameNode allows the user to start writing data.
- Data is written to the data queue and a data streamer sends it to the DataNodes for storage and replication.

3. How does HDFS ensure fault tolerance and high availability?

Refer Question 1

4. What is the function of the NameNode and secondary NameNode in HDFS?

- **NameNode**:
    - It is the master server in HDFS that manages the namespace and regulates access to files by clients.
    - It maintains and manages the file system namespace and metadata for all the files and directories.
    - This includes the filesystem tree, the metadata for all the files and directories, and the list of blocks which defines the data.
- **Secondary NameNode**:
    - It works alongside the NameNode to prevent data loss.
    - It periodically merges the changes (edits) with the filesystem image, kept in memory by the NameNode, and writes it back to the disk.
    - This process is known as a checkpoint.
    - The Secondary NameNode is not a backup to the NameNode but rather assists in reducing the size of the list of file system actions, the Edit Log, which the NameNode keeps in memory.

5. Discuss the strengths and drawbacks of RESTful web services.

Strengths:

- **Scalability -** RESTful APIs are scalable because the server and client are separate. This allows for independent development projects and makes it easier to move data from one server to another.
- **Flexibility -** Uses standard HTTP verbs like GET, POST, PUT, and DELETE.
- **Cacheability:** Utilizes HTTP caching mechanisms for improved performance.
- **Ease of use -** Simple and easy to use, even for those without web development experience.
- **Statelessness -** simplifies server side logic and promotes clear separation of concerns.

Drawbacks:

- **Limited Functionality**: Primarily focused on CRUD operations (Create, Read, Update, Delete).
- **Over/Under-fetching**: May lead to inefficient data transfer.
- **Versioning Complexity**: Challenges in maintaining backward compatibility.
- **Performance Overhead**: HTTP introduces latency and bandwidth overhead.
- **Security Concerns**: Requires careful implementation of security features to mitigate risks.

6. Describe the role of a distributed file system in a job execution environment such as MapReduce in a large-scale cloud system.

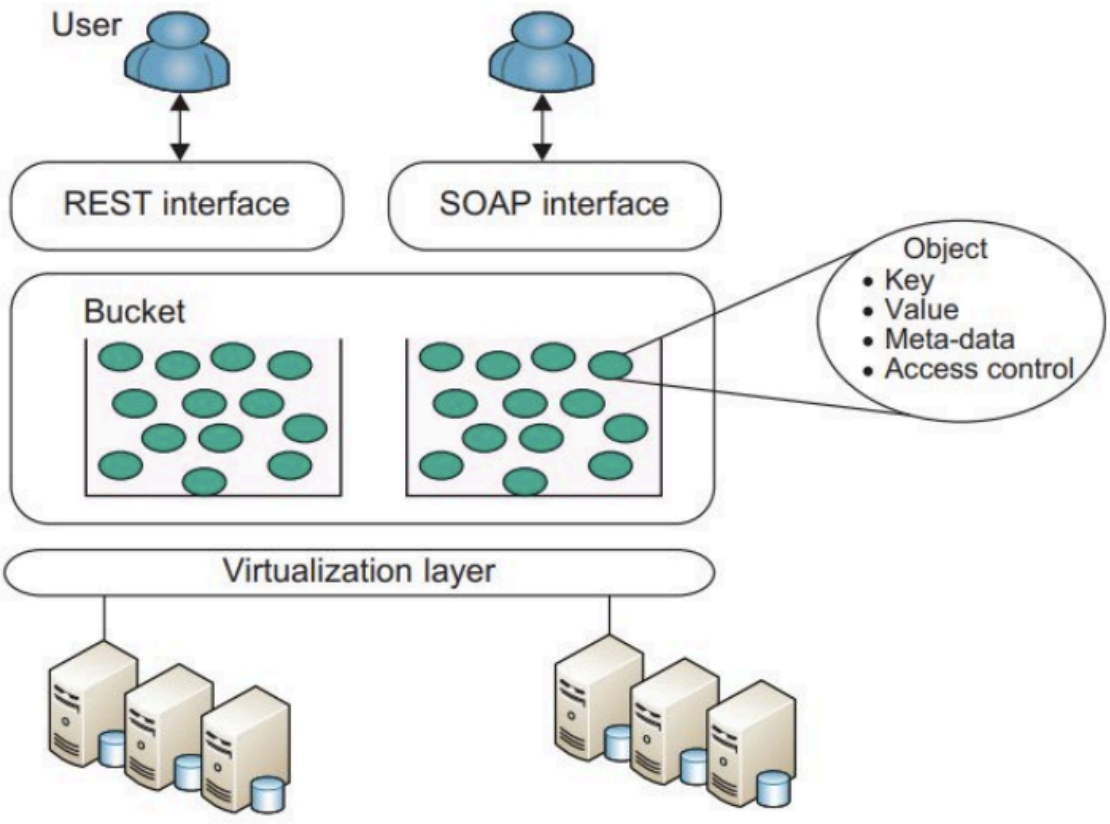
In a large-scale cloud system, a distributed file system plays a crucial role in supporting job execution environments like MapReduce. Here's how:

- **Storage of Input Data**: Distributed file systems store massive volumes of input data required for processing by job execution frameworks like MapReduce. This data can include structured or unstructured data from various sources such as logs, databases, or sensor feeds.
- **Data Locality:** Distributed file systems are designed to provide data locality, ensuring that data is stored close to the computation nodes where processing will occur. This minimizes network overhead by reducing data transfer between storage and processing nodes, thus improving performance.
- **Scalability:** Cloud-based distributed file systems are inherently scalable, allowing them to accommodate the storage needs of large-scale job execution environments. As the volume of data grows or the computational demands increase, distributed file systems can scale out horizontally by adding more storage nodes without disrupting ongoing operations.
- **Fault Tolerance:** Distributed file systems incorporate mechanisms for fault tolerance to ensure data durability and system reliability. Techniques like data replication across multiple nodes and automatic recovery from node failures help maintain data integrity and availability during job execution.
- **Efficient Data Access:** Distributed file systems optimize data access patterns for job execution frameworks like MapReduce. They support parallel access to data across multiple nodes, enabling efficient data processing by distributing computation tasks among available resources.
- **Integration with Job Execution Frameworks:** Distributed file systems are tightly integrated with job execution frameworks like MapReduce. These frameworks leverage the distributed file system's capabilities for data storage, retrieval, and processing, enabling efficient and scalable execution of complex data processing tasks.

7. How are S3 and EBS instances of AWS helpful for any cloud application?

Amazon S3 (Simple Storage Service):

- Provides the object-oriented storage service for users.
- Users can access their objects through Simple Object Access Protocol (SOAP).
- The fundamental operation unit of S3 is called an object.
- Each object is stored in a bucket and retrieved via a unique, developer-assigned key.
- Other attributes - values, metadata, and access control information.
- Two types of web service interface - REST & SOAP.



Amazon S3 Execution Environment

Amazon EBS (Elastic Block Store):

- EBS provides the volume block interface for saving and restoring the virtual images of EC2 instances.
- Users can use EBS to save persistent data and mount to the running instances of EC2.
- EBS is analogous to a distributed file system accessed by traditional OS disk access mechanisms.
- EBS allows you to create storage volumes from 1 GB to 1 TB that can be mounted as EC2 instances
- Multiple volumes can be mounted to the same instance.
- Snapshots are provided so that the data can be saved incrementally.

8. How has publish subscribe middleware been incorporated into the Database system?

The document discusses two main approaches for incorporating publish-subscribe middleware into database systems:

1. Extending Existing Database Systems:

- This approach involves adding publish-subscribe functionalities to existing open-source or commercial database systems.
- Examples include:
    - **PostgreSQL:** Extended with publish-subscribe middleware by Jean Bacon et al. This extension leverages active databases and the publish-subscribe model to create a global event-based system. Databases can define and advertise changes as events, and clients can subscribe to these events with content-based filters. This allows local databases to act as event brokers, routing events among publishers, subscribers, and other brokers.

2. Utilizing Messaging Features of Database Systems:

- Some database systems inherently provide messaging features that can be used for a publish-subscribe approach.
- Examples include:
    - **Oracle Advanced Queuing:** Introduced in Oracle 8i and enhanced in later versions. It offers publish-subscribe functionality built upon message queuing. Features include rule-based subscribers, message propagation, and notification capabilities. It leverages Oracle Streams for persistent message storage, propagation, and transmission. This approach benefits from existing database functionalities like HA, scalability, and security.

Both approaches highlight the advantages of integrating publish-subscribe with databases:

- **Simplified Information Management:** Security, configuration, and recovery tasks for both databases and pub/sub operations can be managed under a single interface.
- **Improved Efficiency:** Data changes can be published, and interested applications can be notified, eliminating the need for inefficient periodic polling.

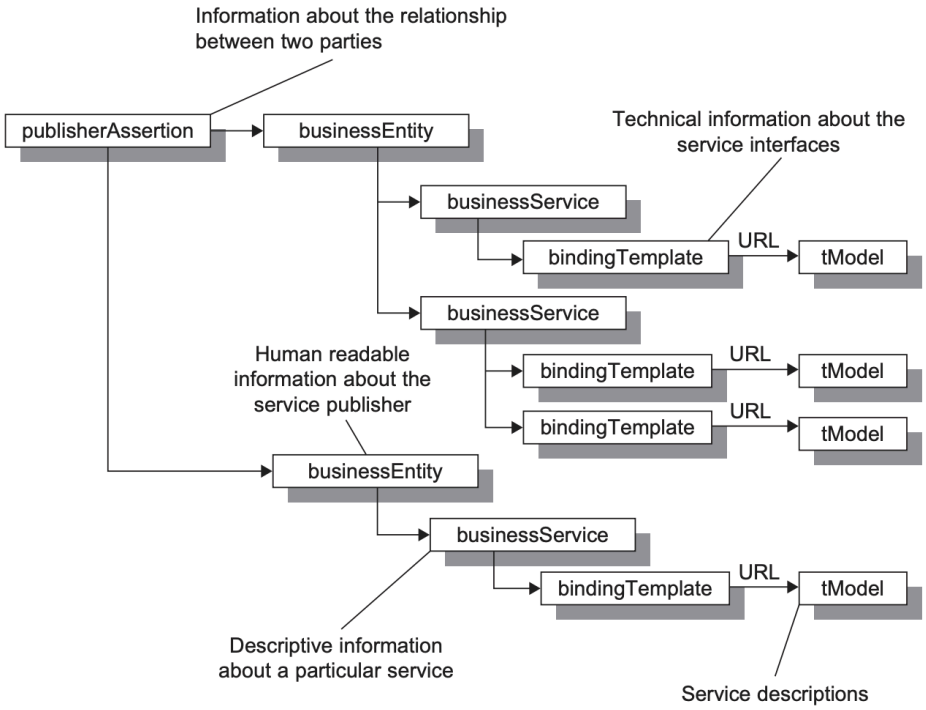
9. What are the three main categories of information that a service registry contains? Which entities map these categories of information in UDDI?

Registries usually contain three categories of information:

- **White pages** contain name and general contact information about an entity.
- **Yellow pages** contain classification information about the types and location of the services the entry offers.
- **Green pages** contain information about the details of how to invoke the offered services.

The entities businessEntity, businessService, bindingTemplate, and tModel form the core data structures of UDDI, each of which can be uniquely identified and accessed by a URI, called the “UDDI key.”

- **businessEntity:** Describes an organization or a business that provides the web services, including the company name, contact information, industry/product/geographic classification, and so on.
- **businessService:** Describes a collection of related instances of web services offered by an organization, such as the name of the service, a description, and so forth.
- **bindingTemplate:** Describes the technical information necessary to use a particular web service, such as the URL address to access the web service instance and references to its description.
- **tModel:** A generic container for specification of WSDL documents in general web services.
- **publisherAssertion:** Defines a relationship between two or more businessEntity elements.
- **subscription:** A standing request to keep track of changes to the entities in the subscription.



10. Daemon processes of HDFS and map reduce.

HDFS

NameNode

The NameNode Daemon is used over the Master System. The management of all MetaData is the primary responsibility of Namenode.

DataNode

The DataNode system is built on Slave-based architecture. On the slave system, DataNode is a program that reads and writes data in response to a client's request.

Secondary NameNode

Backups of the primary NameNode's data are performed every hour on the secondary NameNode. The Hadoop cluster's secondary Namenode will create backups or checkpoints of the Data. This file will be used if the Hadoop cluster fails or crashes.

MapReduce

JobTracker

JobTracker's principal function is Resource Management, which encompasses tracking TaskTrackers, monitoring their progress, and fault tolerance. When a customer sends a job to JobTracker, that job gets broken down into its parts and assigned jobs. After that, the JobTracker decides which jobs should be assigned to each worker node.

TaskTracker

TaskTracker is a daemon that runs MapReduce. It is a program used by slave nodes. The TaskTracker is accountable for completing every one of the responsibilities delegated to them by the JobTracker.

11. Write map and reduce methods to perform a ‘grep’ to find occurrences of a given string (case-sensitive) in a file. it may occur more than once in a given line. The output should use the desired string as a key and give a list of all the line numbers (as Integer) on which the string occurs. If a word occurs multiple times in a line, the line number should be output multiple times. ((the (7 46 52 63 72 73 85 94 116 168 172 182 184))). Draw the execution workflow of the above task in the MapReduce programming environment.

Input:

- File path
- Search string (case-sensitive)

Output:

- Dictionary where key is the search string and value is a sorted list of line numbers where the string appeared.

Map Phase:

➔ **For each line in the file:**
- ◆ Split the line into words.
- ◆ **For each word in the line:**
  - If the word is equal to the search string:
    - Emit key-value pair: (word, line number)

Shuffle and Sort:

➔ Shuffle all key-value pairs generated by all map tasks.
➔ Sort all key-value pairs by key (word).

Reduce Phase:

➔ **For each unique key (word):**
- ◆ Initialize an empty list `line_numbers`.
- ◆ **For each value (line number) associated with the key:**

- Append the line number to `line_numbers`.

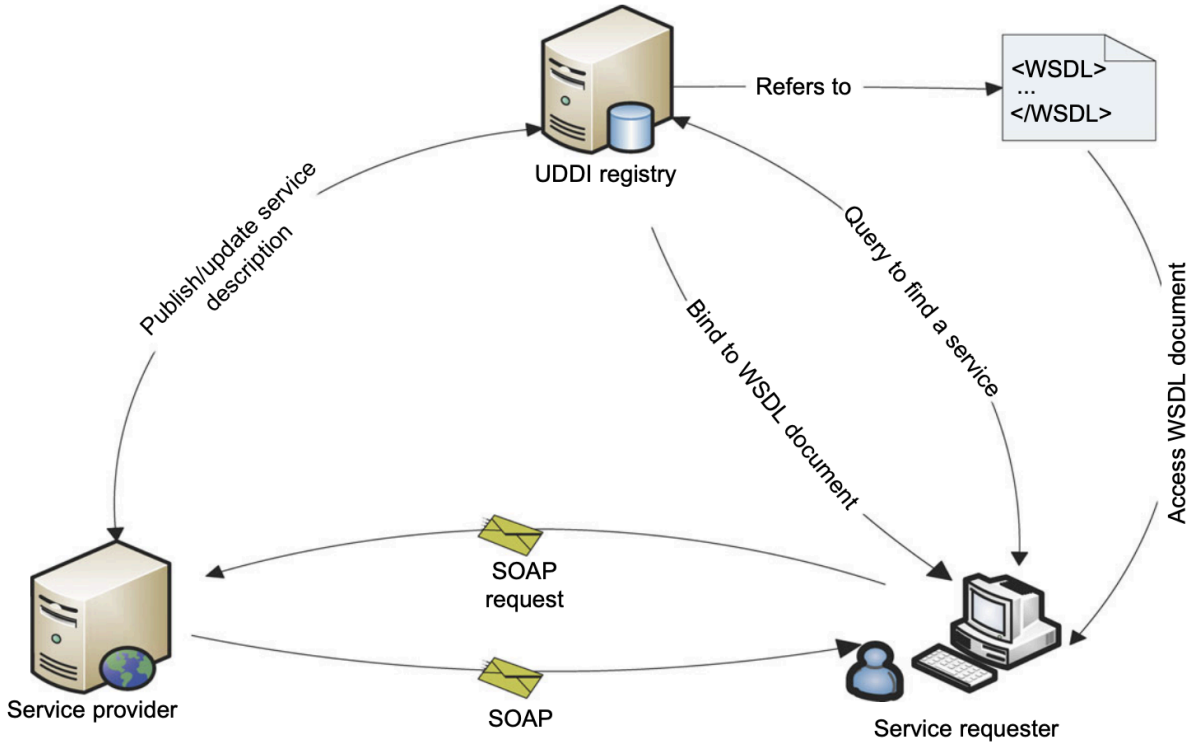
◆ Sort `line_numbers`.

◆ Emit key-value pair: (word, sorted(line_numbers))

Output:

- Print the final key-value pairs generated by the reduce phase. This represents the search string and a sorted list of line numbers where it appeared.

12. Describe how an deployed web service is interacting with other applications and web services using a neat diagram. Discuss the technologies used in this process.



The interaction of a deployed web service with other applications and web services involves several key technologies:

SOAP (Simple Object Access Protocol):

- This protocol is used for transmitting XML documents over Internet protocols like HTTP, SMTP, and FTP.
- It allows for interoperability between different middleware systems through a standard message format.

WSDL (Web Services Description Language):

- WSDL is used to describe the interface of a web service in a standard format.
- It specifies the operations supported by the web service, the input and output parameters, and the protocol binding for message transfer.

UDDI (Universal Description, Discovery, and Integration):

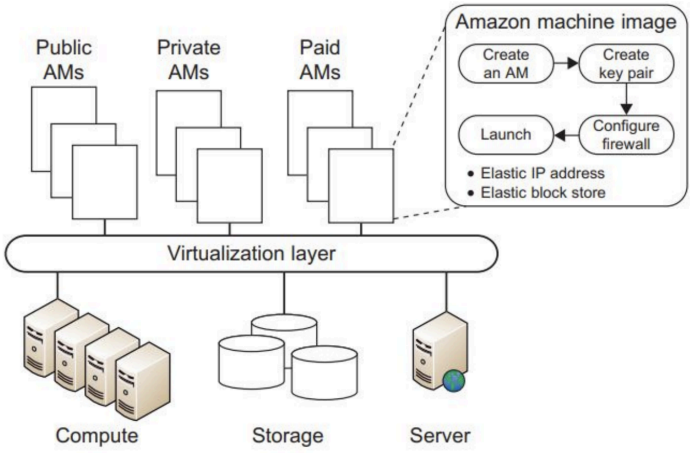
- UDDI acts as a global registry for advertising and discovering web services.
- It enables searching for web services based on names, identifiers, categories, or specifications.

13. Explain the different types of message oriented middleware for supporting distributed computing.

- **Asynchronous Communication**: This asynchronous communication pattern helps improve system responsiveness and scalability.
- **Reliability and Fault Tolerance**: Message queues provided by MOM systems often include features such as message persistence, acknowledgments, and message redelivery, ensuring reliable message delivery even in the presence of failures or network issues. This enhances the fault tolerance and resilience of cloud-based applications.
- **Scalability**: Cloud-based MOM solutions can dynamically allocate resources to handle varying message loads, making them well-suited for cloud environments where scalability is essential.
- **Loose Coupling**: This allows for greater flexibility in system design, deployment, and maintenance, as components can be modified or replaced without affecting the overall system architecture.

- **Integration and Interoperability**: MOM systems support various messaging protocols and standards, enabling seamless integration and interoperability between different technologies, programming languages, and platforms. This flexibility is particularly valuable in heterogeneous cloud environments where multiple technologies may be used.
- **Message Transformation and Routing**: MOM systems often provide features for message transformation and routing, allowing messages to be transformed from one format to another and routed to the appropriate destination based on predefined rules.

14. Explain Amazon EC2 execution environment.



- First to introduce VMs in application hosting.
- Elastic feature - customers can use server instances as needed, paying by the hour for active servers.
- Instances are called Amazon Machine Images (AMIs) which are preconfigured with operating systems based on Linux or Windows, and additional software.
- AMIs are the templates for instances, which are running VMs.
- Workflow to create a VM:

  Create an AMI → Create key pair → Configure firewall → Launch

15. Explain Dryad architecture.

The architecture of Dryad is focused on distributed data-parallel programs and is structured as follows:

- **DAG Structure**: Dryad programs are defined by a directed acyclic graph (DAG) where vertices represent computation tasks and edges represent data channels between these tasks.
- **Job Manager**: A Dryad job is controlled by a job manager, which is responsible for deploying the program across multiple nodes in a cluster.
- **Name Server**: The system includes a name server to enumerate available computing resources, aiding the job manager in scheduling decisions.
- **Execution**: During runtime, a lightweight daemon runs on each cluster node to execute assigned tasks, with communication between nodes monitored by the job manager. It uses a 2D distributed set of pipes, allowing simultaneous processing of large-scale data.

This architecture allows for flexible data flow and parallel execution of large-scale data processing tasks.

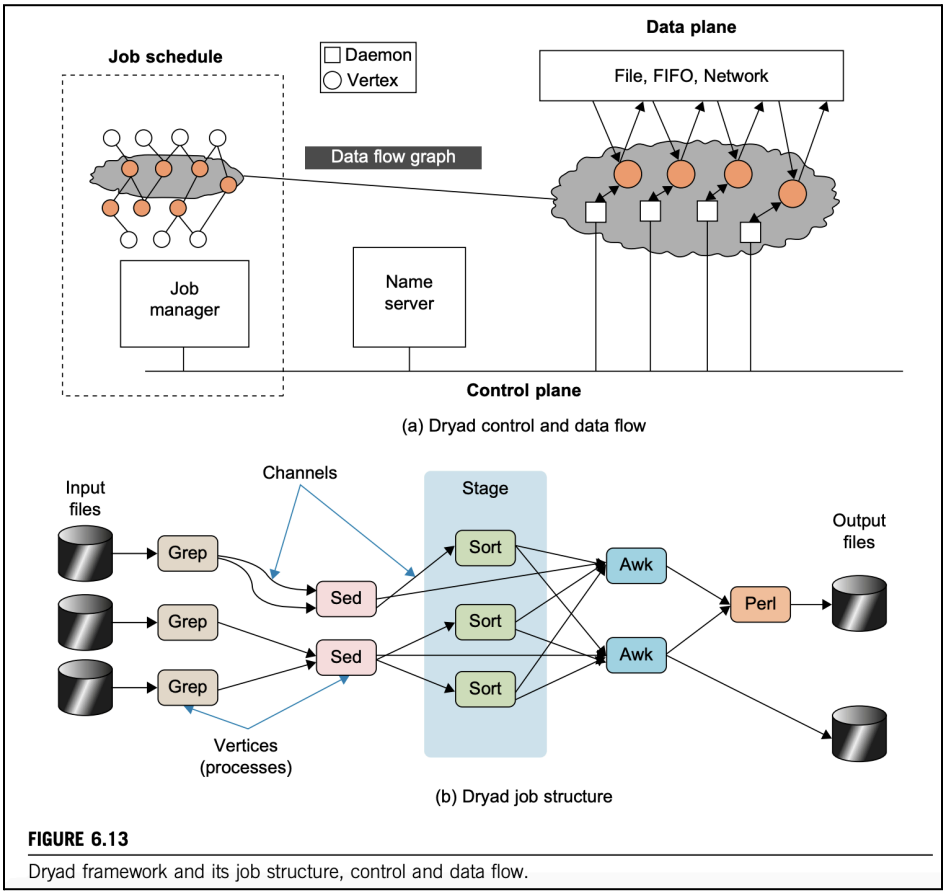


FIGURE 6.13

Dryad framework and its job structure, control and data flow.

P2P and grid

16. Functional modules of globus toolkit

Service Functionality	Module Name	Functional Description
Global Resource Allocation Manager	GRAM	Grid Resource Access and Management (HTTP-based)
Communication	Nexus	Unicast and multicast communication
Grid Security Infrastructure	GSI	Authentication and related security services
Monitory and Discovery Service	MDS	Distributed access to structure and state information
Health and Status	HBM	Heartbeat monitoring of system components
Global Access of Secondary Storage	GASS	Grid access of data in remote secondary storage
Grid File Transfer	GridFTP	Inter-node fast file transfer

17. four access models for organizing a data grid

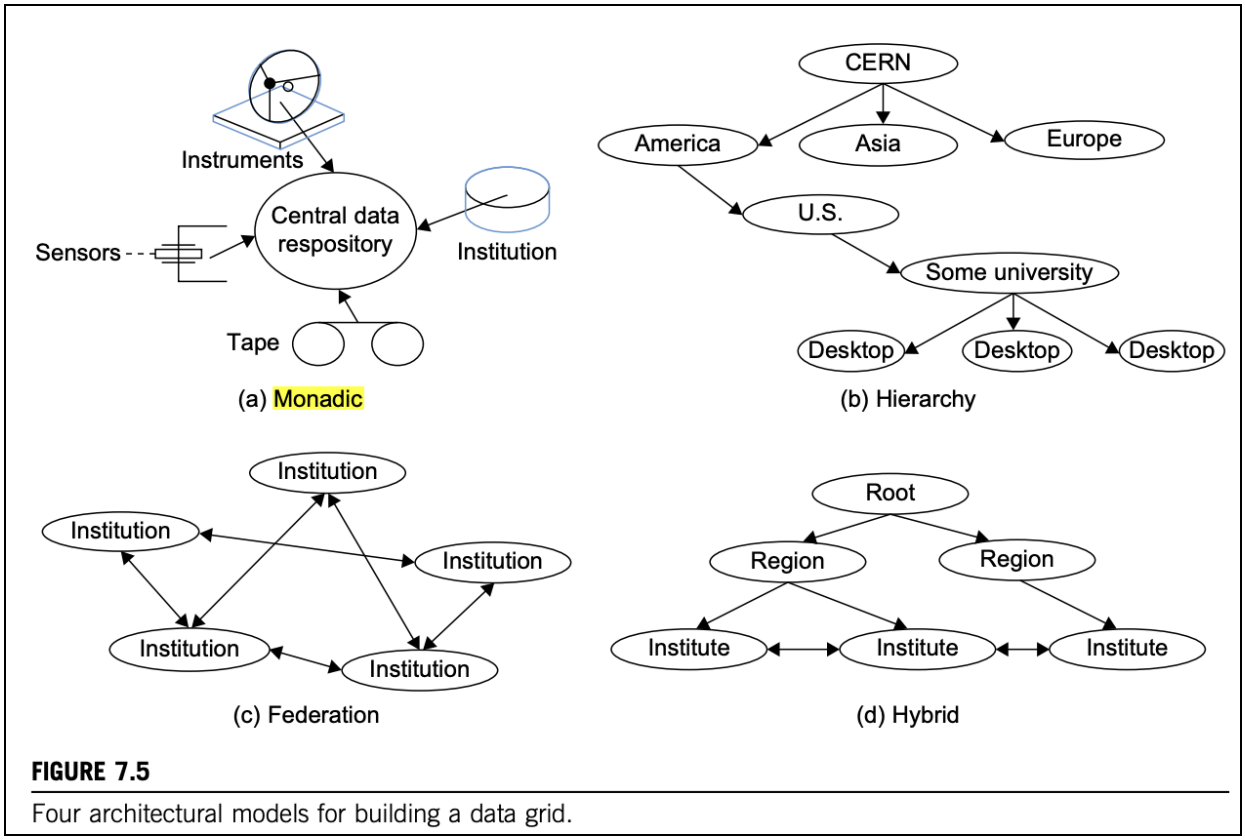


FIGURE 7.5

Four architectural models for building a data grid.

Monadic:

- In the monadic access model, data is organized in a single, centralized grid structure.
- All data storage and processing resources are controlled by a single administrative domain.
- This model offers simplicity and centralized management but may lead to scalability and performance limitations as the size of the data grid grows.

Hierarchical:

- The hierarchical access model organizes data in a hierarchical structure with multiple levels of grids or domains.
- Each level may have its own administrative domain responsible for managing resources within that level.

- Data is distributed across multiple grids, with higher-level grids coordinating and managing lower-level grids.
- This model offers scalability and flexibility by allowing independent management of each grid level, but it may introduce complexity in coordination and communication between levels.

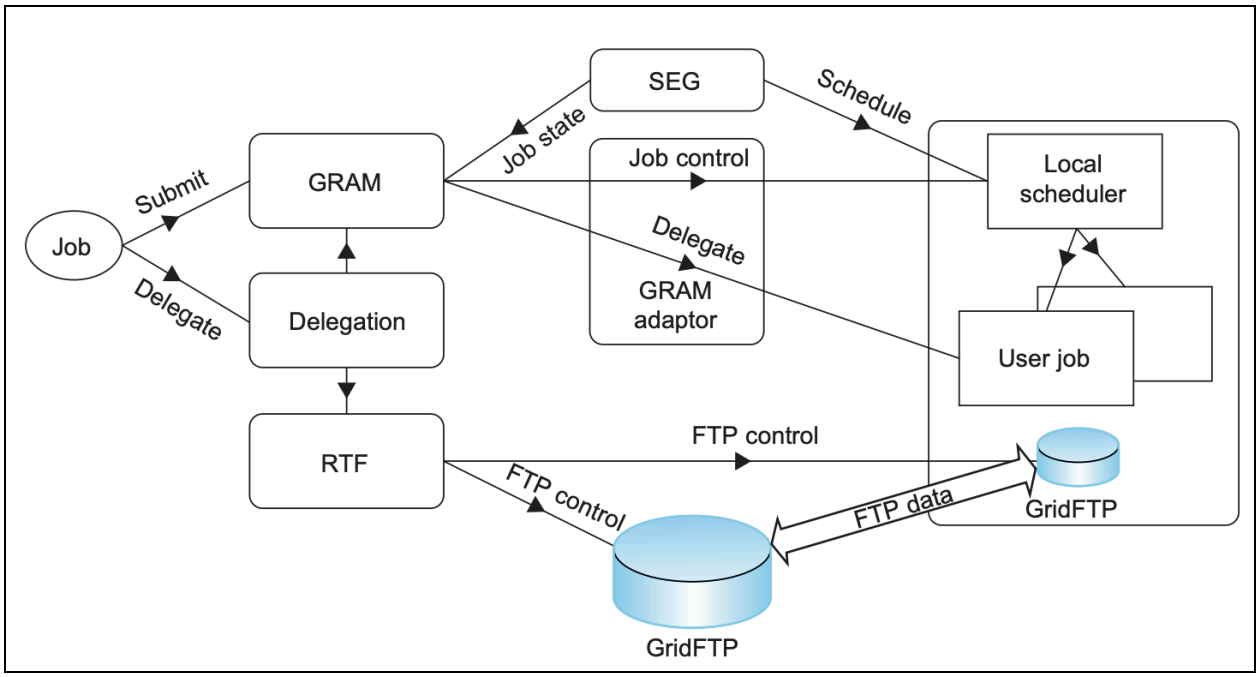
Federated:

- In the federated access model, data grids are distributed across multiple administrative domains or organizations.
- Each domain maintains control over its resources and data, while still participating in a larger federated grid.
- Data sharing and access control mechanisms are established to enable collaboration and interoperability between domains.
- This model promotes decentralization and collaboration but requires strong coordination and governance mechanisms to ensure data consistency and security across federated domains.

Hybrid:

- The hybrid access model combines elements of multiple access models, such as monadic, hierarchical, and federated, to address specific requirements or constraints.
- For example, a hybrid model may involve a combination of centralized and distributed grids, or a mix of hierarchical levels with federated domains.
- This model offers flexibility to adapt to diverse organizational structures, data management needs, and scalability requirements.

18. Globus job workflow among interactive functional modules.



The Globus job workflow involves several interactive functional modules that manage the execution of computational jobs on remote computers in a grid environment.

- **Delegation**: The user delegates their credentials to a delegation service.
- **Job Submission**: The user submits a job request to GRAM (Global Resource Allocation Manager) with the delegation identifier.
- **File Transfer**: GRAM sends a transfer request to RFT (Reliable File Transfer), which uses GridFTP to transfer necessary files.
- **Job Execution**: GRAM invokes a local scheduler via a GRAM adapter, and the SEG (Scheduler Event Generator) initiates the user jobs.
- **Monitoring**: The local scheduler reports the job state to the SEG, and upon completion, GRAM stages out the resultant files using RFT and GridFTP.
- **Notification**: The grid monitors the progress and sends notifications to the user about the job status.

19. Distinction between Client/Server and P2P Architectures

Client-Server Network	Peer-to-Peer Network
Differentiated clients and server	No differentiation between clients and server
Focuses on information sharing	Focuses on connectivity
Uses centralized server for data storage	Each peer has its own data
Server responds to client requests	Nodes request and respond to services
More costly than Peer-to-Peer Networks	Less costly than Client-Server Networks
More stable, especially with increasing clients	Less stable with more peers
Suitable for both small and large networks	Generally suited for small networks (less than 10 computers)

20. Summarize the common characteristics of p2p networks.

- **Decentralization**: P2P networks operate in a decentralized manner where each node can act as both a client and a server, enabling direct communication and resource sharing between peers without the need for a central server.
- **Scalability**: P2P networks are inherently scalable as the addition of new nodes does not significantly impact the overall network performance. This scalability is achieved through the distribution of resources and responsibilities among peers.
- **Self-organization**: P2P networks are self-organizing systems where peers autonomously join and leave the network without disrupting its operation. This **self-organization** helps in adapting to changes in network conditions and peer availability.

- **Resource Sharing**: Peers in a P2P network share resources such as files, processing power, and bandwidth with other peers in a cooperative manner. This sharing of resources allows for efficient utilization of network resources.
- **Fault Tolerance**: P2P networks exhibit fault tolerance by distributing data and services across multiple nodes. If a node fails or leaves the network, other nodes can still access the shared resources, ensuring continuous operation.
- **Data Replication**: To improve data availability and access speed, P2P networks often employ data replication techniques where copies of data are stored on multiple nodes. This redundancy helps in mitigating data loss and improving data retrieval performance.
- **Security and Trust**: P2P networks face challenges related to security and trust due to the distributed nature of the network. Various trust models and reputation systems are studied to establish trust among peers and ensure secure interactions within the network.
- **Content Distribution**: P2P networks are commonly used for content distribution, including file sharing, video streaming, and software distribution. Different P2P content distribution technologies like BitTorrent and eMule employ efficient mechanisms for distributing content among peers.

21. Distributed hash table (DHT) is middleware which offers information search or table lookup services for a distributed system. Explain.

1. Key-Value Pairs and Identifier Space:

- DHT stores data as (key, value) pairs, similar to a regular hash table.
- Keys are mapped to the identifier space using a hash function. For example, an identifier space allowing 64 bit binary strings can accommodate 2^{64} keys.

2. Distributed Ownership and Peer Addressing:

- The identifier space ownership is distributed among the participating nodes in the P2P network.
- Each peer's identifier is a hash value of its address, created using a function like SHA-1.

3. Lookup Service and Minimal Disruption:

- DHT allows any peer to search for a value associated with a specific key.
- The mapping of keys to values is spread across the nodes, ensuring minimal disruption when a node joins or leaves the network.

4. Scalability and Consistent Hashing:

- DHT design is efficient and scales well to extremely large identifier spaces.
- It utilizes consistent hashing to map keys to the "closest" nodes based on their identifier space proximity.

In summary, DHT acts like a distributed database where keys and values are spread across the network. It provides a lookup service for any peer to efficiently find the data associated with a key, even with frequent node arrivals and departures.

22. Define blind flooding with respect to Unstructured P2P overlay networks.

- Unstructured P2P overlay networks give no indication of how to locate a specified node since node neighborhoods are randomly selected and are not subject to constraints.
- The routing algorithms in unstructured P2P overlay networks are always based on **blind flooding**.
- When receiving a message, a peer node simply forwards it to all its neighbors except the one from which it receives the message.
- Suppose the number of nodes is n and the average node degree (i.e., the number of neighbors) is k.
- The number of messages used for locating a node is *n(k-1)* on average.

23. The three types of approaches to exploit network proximity in structured P2P overlay.

Geographic Layout:

- Assigns identifiers based on physical location. Nearby nodes get closer identifiers in the overlay network.
- Only suitable for specific overlays like Content-Addressable Network (CAN).
- Landmark binning scheme:
    - Uses a set of landmark nodes.
    - Each node measures distance to landmarks and assigns itself to a "bin" based on closest landmarks.
    - CAN space is divided into zones based on bins.
    - Nodes pick a random point within their zone to join the network.
    - Increases proximity but can overload specific zones.

Proximity Routing:

- Chooses the path with the least delay for message forwarding between nodes.
- Finding optimal paths is complex (NP-hard).
- Heuristic approach: forward to the closest neighbor.
- May increase routing path length.
- Well suited for Pastry networks.

Proximity Neighbor Selection:

- Nodes prioritize physically close neighbors in the overlay network, while respecting identifier constraints.
- Example in Chord network: choosing the closest successor within a specific range.
- Landmark clusters with dimensionality reduction:
    - Measures distance to landmarks to create a position vector.
    - Uses space-filling curves to map high-dimensional data to lower dimensions while preserving proximity.
    - Limited accuracy for very close nodes, can be improved with Round-Trip Time (RTT) measurement.
- Easier to optimize in unstructured overlays due to more flexible neighbor relationships.

Each of these approaches leverages network proximity information to enhance the efficiency and performance of structured P2P overlay networks. By exploiting proximity-aware techniques in neighbor selection, routing optimization, and data replication/placement, these networks can achieve better scalability, reliability, and responsiveness, making them suitable for a wide range of distributed applications and services.