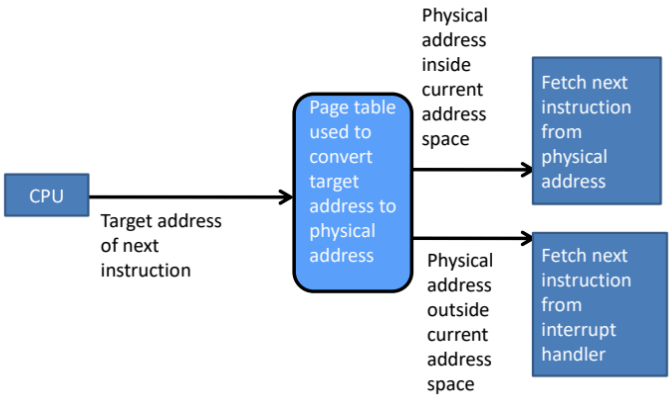
Architectures for the cloud Overview 2

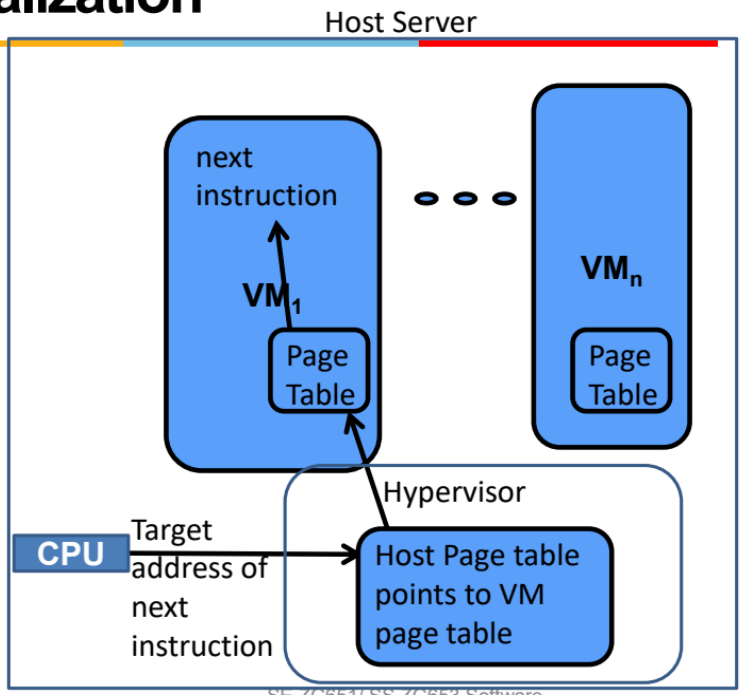
Virtual Memory Page Table

Virtual Memory Page Table

Virtual memory for non-virtualized application

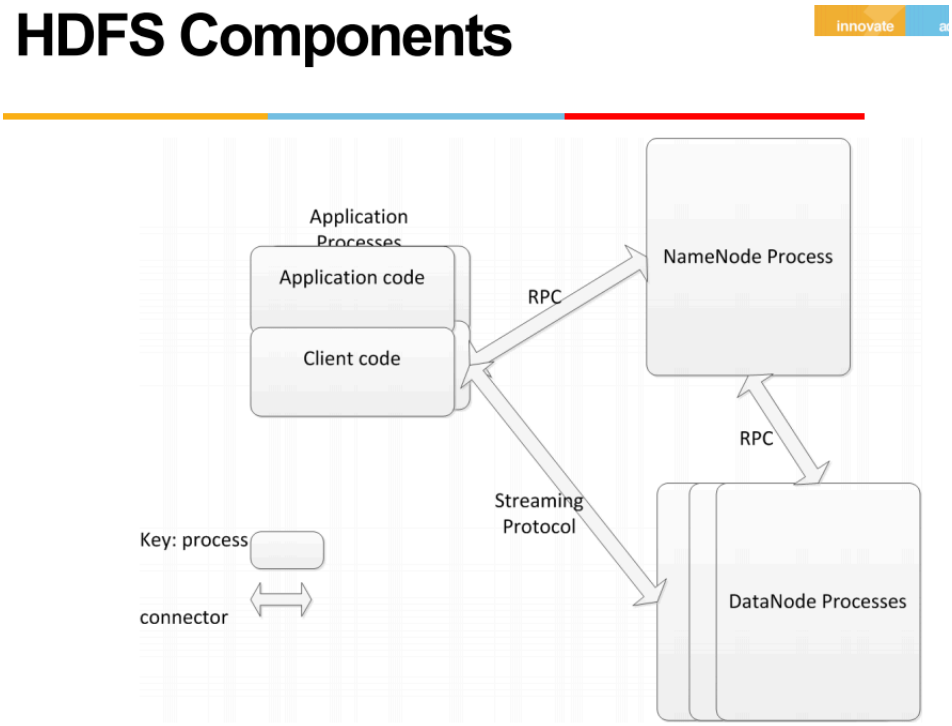


Hypervisor Manages Virtualization



1. Base Mechanisms

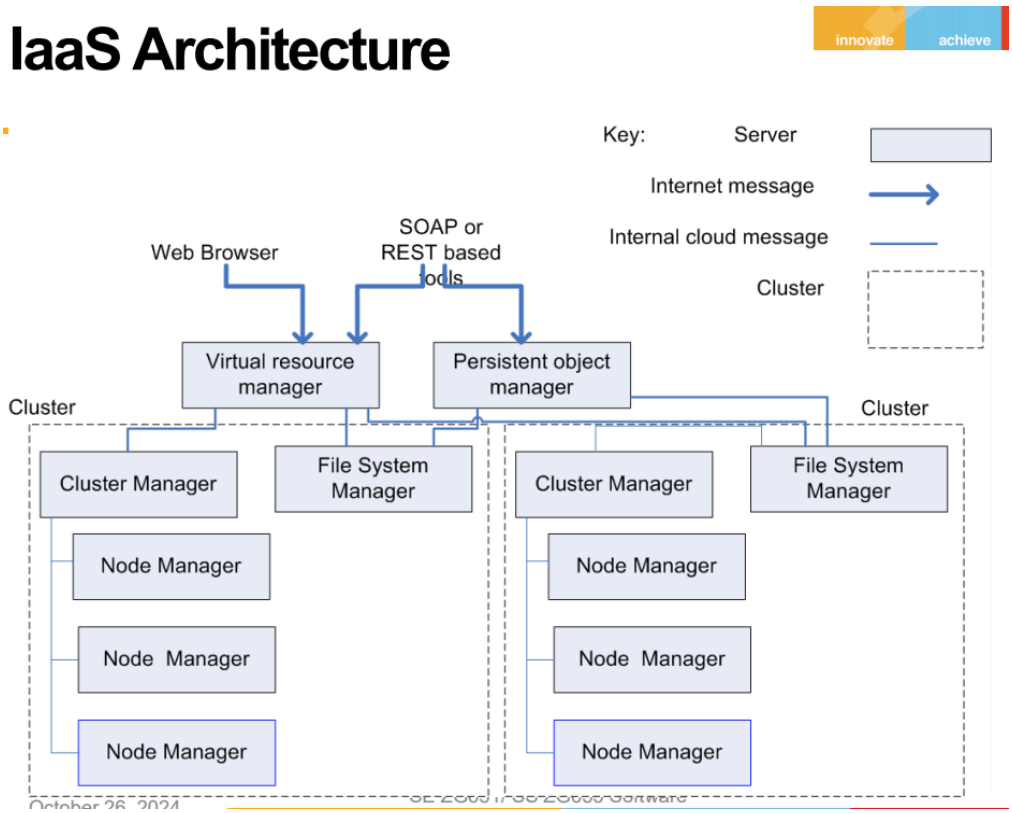
- **Hypervisor**:
  - The core of cloud infrastructure, enabling virtualization by managing multiple virtual machines (VMs) on a single physical server.
  - **Use Case**: VMware ESXi or Microsoft Hyper-V as hypervisors that allocate resources among different VMs running on the same server.
- **Virtual Machine (VM)**:
  - A software emulation of a physical computer, isolating each VM's address space.
  - VMs appear as independent machines to applications, have their own IP addresses, and can run various operating systems.
  - **Diagram**: Illustrates how VMs interact with the hypervisor on a host server.
  - **Use Case**: Deploying multiple VMs on a single server to run different services like web servers, databases, and app servers concurrently.
- **File System**:
  - Cloud file systems ensure persistent storage, often using distributed systems like Hadoop Distributed File System (HDFS).
  - **HDFS Write - Sunny Day Scenario**:
    - The client writes a block to a DataNode, which replicates it to additional DataNodes for redundancy.
    - **Failure Scenarios**: Handles client, NameNode, or DataNode failures, ensuring data persistence via replication and retries.
  - **Diagram**:



- **Network**:
  - Every VM is assigned an IP address, enabling communication via standard TCP/IP protocols.
  - Gateways adjust IP addresses for communication management.
  - **Use Case**: Cloud applications dynamically adjusting network configurations to maintain communication efficiency.

2. Sample Technologies

- **Infrastructure as a Service (IaaS)**:
  - Provides hardware resources like servers, storage, and networking as virtualized components.
  - **Architecture Components**:
    - **Cluster Manager**: Manages clusters of servers.
    - **Persistent Object Manager**: Manages persistent storage.
    - **Virtual Resource Manager**: Acts as a gateway for resource allocation and messaging.
    - **File System Manager**: Manages network-wide file storage, similar to HDFS.
  - **Services Provided**:
    - **Automatic IP Reallocation**: Handles IP changes in case of VM failure.
    - **Automatic Scaling**: Adjusts the number of VMs based on demand.
  - **Diagram**:



    - **Use Case**: AWS EC2 or Microsoft Azure managing cloud infrastructure for scalable applications.
- **Platform as a Service (PaaS)**:
    - Provides an integrated stack for development, e.g., LAMP stack (Linux, Apache, MySQL, Python).
    - **Use Case**: Developers build web applications on Heroku or Google App Engine, relying on PaaS to manage infrastructure, databases, and middleware.
- **Databases**:
    - **Why Relational Databases Fell Short**:
        - Challenges with massive web data processing, the CAP theorem's constraints, and relational models' limitations in handling dynamic data.
        - New models emerged:
            - **Key-Value Databases (e.g., HBase)**:
                - Uses keys to access values without schemas; time stamps detect collisions.
            - **Document-Centric Databases (e.g., MongoDB)**:
                - Stores data as objects, with flexible schemas and eventual consistency.
    - **What is Omitted from These Databases**:
        - Transactions, normalization, and strict consistency are often compromised for scalability.

3. Architecting in a Cloud Environment

- **Security**:
    - Multi-tenancy raises new security concerns:
        - **Information Sharing Risks**: Possible data leaks from shared resources (e.g., disks).
        - **VM Escape Risks**: Theoretical attacks that break hypervisor isolation.
        - **Side Channel Attacks**: Exploit shared resources like caches to gather information.
        - **Denial of Service (DoS) Attacks**: Resource exhaustion attacks by one user affecting others.
    - **Mitigation Strategies**: Use encryption, strict resource isolation, and proactive monitoring.
    - **Use Case**: Banking applications implementing enhanced encryption and resource isolation when hosted on public clouds.
- **Performance**:
    - **Auto-Scaling**: Adjusts resources dynamically based on load.
        - Response times for scaling may not always match demand spikes.
        - **Use Case**: An e-commerce platform scaling servers during flash sales but facing potential latency in adding new resources.
    - **Proactive Resource Management**: Applications should anticipate resource requirements and request them before bottlenecks occur.
- **Availability**:

    - With thousands of servers, failure is expected; cloud providers ensure the infrastructure remains available.
    - Applications should detect and manage instance failures, with built-in recovery mechanisms.
    - **Use Case**: A content delivery network (CDN) ensuring redundancy across multiple data centers to maintain availability during regional outages.

4. Summary

- Cloud architecture requires special attention to virtualization, network management, and distributed storage systems.
- Architecting for the cloud involves considering additional concerns like security, performance, and availability.
- Cloud environments offer benefits like scalability, flexibility, and cost-efficiency, but also introduce challenges in terms of data management, system complexity, and dynamic resource allocation.