Introduction to Database and Cloud Systems

What is a Database?

- Collection of data central to some enterprise
- Essential to operation of enterprise
  - Contains the only record of enterprise activity
- An asset in its own right
  - Historical data can guide enterprise strategy
  - Of interest to other enterprises
- State of database mirrors state of enterprise
  - Database is persistent

What is a Database Management System?

- A Database Management System (DBMS) is a program that manages a database:
  - Supports a high-level access language (e.g. SQL).
  - Application describes database accesses using that language.
  - DBMS interprets statements of language to perform requested database access.

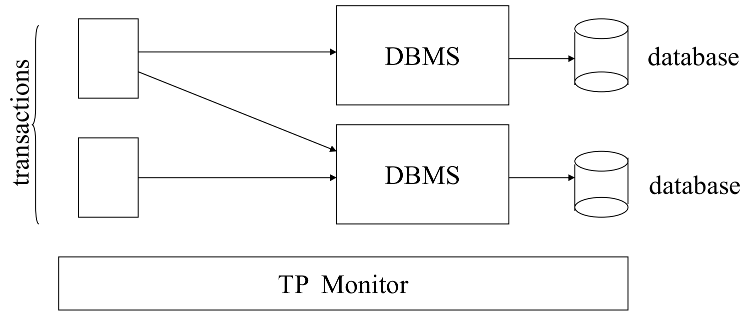
What is a Transaction?

- When an event in the real world changes the state of the enterprise, a transaction is executed to cause the corresponding change in the database state
  - With an on-line database, the event causes the transaction to be executed in real time
- A transaction is an application program with special properties - discussed later - to guarantee it maintains database correctness

What is a Transaction Processing System?

- Transaction execution is controlled by a TP monitor
  - Creates the abstraction of a transaction, analogous to the way an operating system creates the abstraction of a process
  - TP monitor and DBMS together guarantee the special properties of transactions
- A Transaction Processing System consists of TP monitor, databases, and transactions

Transaction Processing System



System Requirements

- **High Availability**: on-line => must be operational while enterprise is functioning
- **High Reliability**: correctly tracks state, does not lose data, controlled concurrency
- **High Throughput**: many users => many transactions/sec
- **Low Response Time**: on-line => users are waiting

System Requirements (con’t)

- **Long Lifetime**: complex systems are not easily replaced
  - Must be designed so they can be easily extended as the needs of the enterprise change
- **Security**: sensitive information must be carefully protected since system is accessible to many users
  - Authentication, authorization, encryption

Roles in Design, Implementation, and Maintenance of a TPS

- **System Analyst** - specifies system using input from customer; provides complete description of functionality from customer's and user's point of view
- **Database Designer** - specifies structure of data that will be stored in database
- **Application Programmer** - implements application programs (transactions) that access data and support enterprise rules

9

9

Roles in Design, Implementation and Maintenance of a TPS (con't)

- **Database Administrator -** maintains database once system is operational: space allocation, performance optimization, database security
- **System Administrator -** maintains transaction processing system: monitors interconnection of HW and SW modules, deals with failures and congestion

10

10

OLTP vs. OLAP

- **On-line Transaction Processing** (OLTP)
  - Day-to-day handling of transactions that result from enterprise operation
  - Maintains correspondence between database state and enterprise state
- **On-line Analytic Processing** (OLAP)
  - Analysis of information in a database for the purpose of making management decisions

11

11

OLAP

- Analyzes historical data (terabytes) using complex queries
- Due to volume of data and complexity of queries, OLAP often uses a data warehouse
- **Data Warehouse -** (offline) repository of historical data generated from OLTP or other sources
- **Data Mining** - use of warehouse data to *discover* relationships that might influence enterprise strategy

12

12

Examples - Supermarket

- OLTP
  - Event is 3 cans of soup and 1 box of crackers bought; update database to reflect that event
- OLAP
  - Last winter in all stores in Ontario, how many customers bought soup and crackers together?
- Data Mining
  - Are there any interesting combinations of foods that customers frequently bought together?

13

13

The Big Picture: Databases

- We are particularly interested [at this stage] in - a 15 minutes review of relational databases
- Data is stored in tables.

14

14

Table

- Set of rows (no duplicates)
- Each *row* describes a different entity
- Each *column* states a particular fact about each entity
  - Each column has an associated *domain*

Id	*Name*	*Address*	*Status*
1111	John	123 Main	fresh
2222	Mary	321 Oak	soph
1234	Bob	444 Pine	soph
9999	Joan	777 Grand	senior

- Domain of *Status* = {fresh, soph, junior, senior}

15

15

Relation

- Mathematical entity corresponding to a table
  - row ~ *tuple*
  - column ~ *attribute*
- Values in a tuple are *related* to each other
  - John lives at 123 Main
- Relation **R** can be thought of as predicate **R**
  - **R**(x,y,z) is true iff tuple (x,y,z) is in **R**

16

16

Operations

- Operations on relations are precisely defined
  - Take relation(s) as argument, produce new relation as result
  - Unary (*e.g.*, delete certain rows)
  - Binary (*e.g.*, union, Cartesian product)
- Corresponding operations defined on tables as well
- Using mathematical properties, *equivalence* can be decided
  - Important for *query optimization*:

$$op1(T1,op2(T2)) \overset{?}{=} op3(op2(T1),T2)$$

17

17

Structured Query Language: SQL

- Language for manipulating tables
- *Declarative* – Statement specifies *what* needs to be obtained, *not how* it is to be achieved (*e.g.*, how to access data, the order of operations)
- Due to declarativity of SQL, DBMS determines evaluation strategy
  - This greatly simplifies application programs
  - *But DBMS is not infallible*: programmers should have an idea of strategies used by DBMS so they can design better tables, indices, statements, in such a way that DBMS can evaluate statements efficiently

18

18

Structured Query Language (SQL)

```
SELECT <attribute list>
FROM  <table list >
WHERE <condition>
```

- Language for constructing a new table from argument table(s).
  - FROM indicates source tables
  - WHERE indicates which *rows* to retain
    - It acts as a filter
  - SELECT indicates which *columns* to extract from retained rows
    - Projection
- The result is a table.

19

19

Example

```
SELECT Name
FROM  Student
WHERE  Id > 4999
```

Id	*Name*	*Address*	*Status*
1234	John	123 Main	fresh
5522	Mary	77 Pine	senior
9876	Bill	83 Oak	junior

Student

Name
Mary
Bill

Result

20

20

Examples

SELECT *Id*, *Name* FROM Student

SELECT *Id*, *Name* FROM Student
WHERE *Status* = 'senior'

SELECT * FROM Student
WHERE *Status* = 'senior'

result is a table with one column and one row

SELECT COUNT(*) FROM Student
WHERE *Status* = 'senior'

21

21

More Complex Example

- **Goal:** table in which each row names a senior and gives a course taken and grade
- Combines information in two tables:
  - Student: *Id*, *Name*, *Address*, *Status*
  - Transcript: *StudId*, *CrsCode*, *Semester*, *Grade*

SELECT *Name*, *CrsCode*, *Grade*
FROM Student, Transcript
WHERE *StudId* = *Id* AND *Status* = '*senior*'

22

22

Join

SELECT *a1*, *b1*
FROM T1, T2
WHERE *a2* = *b2*

T1

a1	*a2*	*a3*
A	1	xxy
B	17	rst

T2

b1	*b2*
3.2	17
4.8	17

FROM T1, T2
yields:

a1	*a2*	*a3*	*b1*	*b2*
A	1	xxy	3.2	17
A	1	xxy	4.8	17
B	17	rst	3.2	17
B	17	rst	4.8	17

WHERE *a2* = *b2*
yields:

B	17	rst	3.2	17
B	17	rst	4.8	17

SELECT *a1*, *b1*
yields result:

B	3.2
B	4.8

23

23

Modifying Tables

UPDATE *Student*
SET *Status* = 'soph'
WHERE *Id* = 111111111

INSERT INTO Student (*Id*, *Name*, *Address*, *Status*)
VALUES (999999999, 'Bill', '432 Pine', 'senior')

DELETE FROM Student
WHERE *Id* = 111111111

24

24

Creating Tables

```
CREATE TABLE Student (
  Id  INTEGER,
  Name CHAR(20),
  Address  CHAR(50),
  Status  CHAR(10),
  PRIMARY KEY (Id)  )
```

Constraint: explained later

25

25

Transactions

- Many enterprises use databases to store information about their state
  - *E.g.*, balances of all depositors
- The occurrence of a real-world event that changes the enterprise state requires the execution of a program that changes the database state in a corresponding way
  - *E.g.*, balance must be updated when you deposit
- A *transaction* is a program that accesses the database in response to real-world events

26

26

Transactions

- Transactions are not just ordinary programs
- Additional requirements are placed on transactions (and particularly their execution environment) that go beyond the requirements placed on ordinary programs.
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability

  ACID properties

  (explained next)

27

27

Integrity Constraints

- Rules of the enterprise generally limit the occurrence of certain real-world events.
  - Student cannot register for a course if current number of registrants = maximum allowed
- Correspondingly, allowable database states are restricted.
  - *cur_reg <= max_reg*
- These limitations are expressed as *integrity constraints,* which are assertions that must be satisfied by the database state.

28

28

Consistency

- Transaction designer must ensure that

  IF the database is in a state that satisfies all integrity constraints when execution of a transaction is started

  THEN when the transaction completes:
  - All integrity constraints are once again satisfied (constraints can be violated in intermediate states)
  - New database state satisfies specifications of transaction

29

29

Atomicity

- A real-world event either happens or does not happen.
  - Student either registers or does not register.
- Similarly, the system must ensure that either the transaction runs to completion (*commits*) or, if it does not complete, it has no effect at all (*aborts*).
  - This is not true of ordinary programs. A hardware or software failure could leave files partially updated.

30

30

Durability

- The system must ensure that once a transaction commits its effect on the database state is not lost in spite of subsequent failures.
  - Not true of ordinary systems. For example, a media failure after a program terminates could cause the file system to be restored to a state that preceded the execution of the program.

31

31

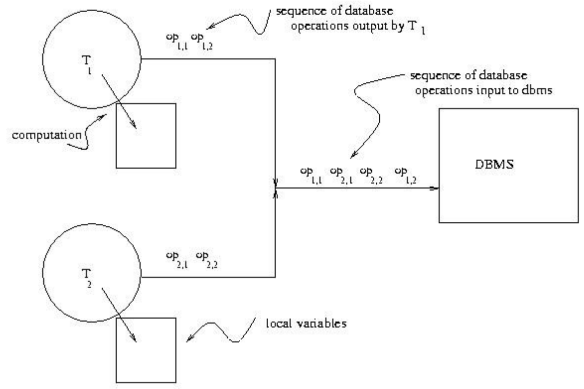
Isolation

- Deals with the execution of multiple transactions concurrently.
- If the initial database state is consistent and accurately reflects the real-world state, then the *serial* (one after another) execution of a set of consistent transactions preserves consistency.
- But serial execution is *inadequate* from a performance perspective.

32

32

Concurrent Transaction Execution


sequence of database operations output by T_1
$op_{1,1}$ $op_{1,2}$
T_1
computation
sequence of database operations input to dbms
$op_{1,1}$ $op_{2,1}$ $op_{2,2}$ $op_{1,2}$
DBMS
T_2
$op_{2,1}$ $op_{2,2}$
local variables

Isolation

- Concurrent (interleaved) execution of a set of transactions offers performance benefits, but might not be correct.
- **Example**: Two students execute the course registration transaction at about the same time

(*cur_reg* is the number of current registrants)

```
T1:  read(cur_reg : 29)                               write(cur_reg : 30)
T2:                      read(cur_reg : 29)  write(cur_reg : 30)
time →
```

Result: Database state no longer corresponds to real-world state, integrity constraint violated.

Isolation

- The effect of concurrently executing a set of transactions must be the same as if they had executed serially in some order
  - The execution is thus *not* serial, but *serializable*
- Serializable execution has better performance than serial, but performance might still be inadequate. Database systems offer several isolation levels with different performance characteristics (but some guarantee correctness only for certain kinds of transactions – not in general)

ACID Properties

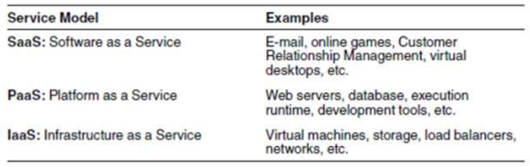
- The transaction monitor is responsible for ensuring atomicity, durability, and (the requested level of) isolation.
  - Hence it provides the abstraction of failure-free, non-concurrent environment, greatly simplifying the task of the transaction designer.
- The transaction designer is responsible for ensuring the consistency of each transaction but doesn't need to worry about concurrency and system failures.

Introduction to Cloud Computing I

- The National Institute of Standards and Technology (NIST) has provided a characterization of the cloud with the following elements:
  - **On-demand self-service**. A consumer can unilaterally provision computing capabilities.
  - **Broad network access**. Capabilities are available over the network
  - **Resource pooling**. The provider's computing resources are pooled to serve multiple consumers.
  - **Rapid elasticity**. Capabilities can be elastically provisioned and released, in some cases automatically.
  - **Measured service**. Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service.

Introduction to Cloud Computing II

- NIST also characterizes the various types of services available from cloud providers, as shown



Service Model	Examples
SaaS: Software as a Service	E-mail, online games, Customer Relationship Management, virtual desktops, etc.
PaaS: Platform as a Service	Web servers, database, execution runtime, development tools, etc.
IaaS: Infrastructure as a Service	Virtual machines, storage, load balancers, networks, etc.

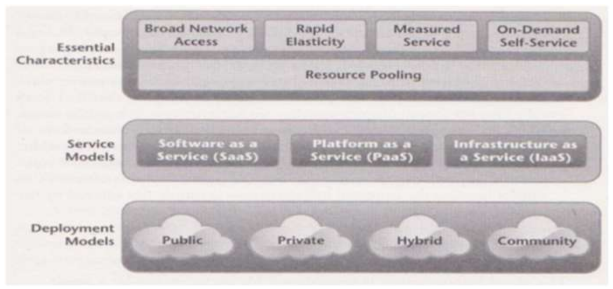
  - **Software as a Service (SaaS).** The consumer is provided the capability to use the provider's applications running on a cloud infrastructure.
  - **Platform as a Service (PaaS).** The consumer is provided the capability to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider.
  - **Infrastructure as a Service (IaaS).** The consumer is provided the capability to provision processing, storage, networks, and other fundamental computing resources

Why cloud?

- In the era of Internet of Things (IoT), there are unprecedented high volume of data generated by the IoT devices that needs to be collected, analyzed, and stored.
- Such high volumes of data can be stored in the cloud with high latency and network bandwidth.
- Technologies used include: **virtual machines** (VMs), **containers**, **fog**, and **edge**.
- Both fog and edge push data analysis closer to the systems or devices that the data originate from.

What is cloud computing?

- Cloud computing concerns the provisioning of resources that include:
  - Computation (i.e. CPU and GPU),
  - Memory,
  - Storage, and
  - Applications/services
- All of these, over the Internet.
- The basic architecture is client/server and centralized deployment, and computation offloading for applications.



Cloud Technology

- Cloud computing is <u>cost-efficient</u> in application deployment and maintenance, and <u>flexible</u> in resource provisioning and it <u>decouple</u> services from underlying technologies at both client and server sides.
- Matured cloud computing platforms and services include: Amazon EC2, Google Cloud Platform, Microsoft Azure, and IBM SmartCloud.
- There are other specialized platforms and services, and mobile cloud computing.

41

41

Application Architecture

- The emergence of data-intensive and highly interconnected distributed applications (e.g. 5G, IoT, big data) are driving the need for new paradigms for the design and deployment of large scale software systems.
- Cloud computing introduces various conceptual models for distributed applications.
- A distributed application in cloud generally consists of a set of components spread over multiple cloud resources.

42

42

Application migration

- Application migration is the relocation of one or multiple components within cloud-DCs.
- Components are generally hosted in virtual machines (VMs) or containers.
- **Migration** implies a VM/container migration.
- **Migration** aims to provide workload balancing, fault tolerance, system stability, and energy efficiency.
- **Migration** can be classified either as live migration or non-live migration.

43

43

Workload Analysis

- Workload is the total number of requests issued by clients to an application or a single application component.
- It can be categorize as:
  - Sequential or non-sequential/random
  - Transactional or non-transactional
  - Computation-intensive
  - Data intensive, or
  - Memory-sensitive
- The demand for resources is different for each type.

44

44

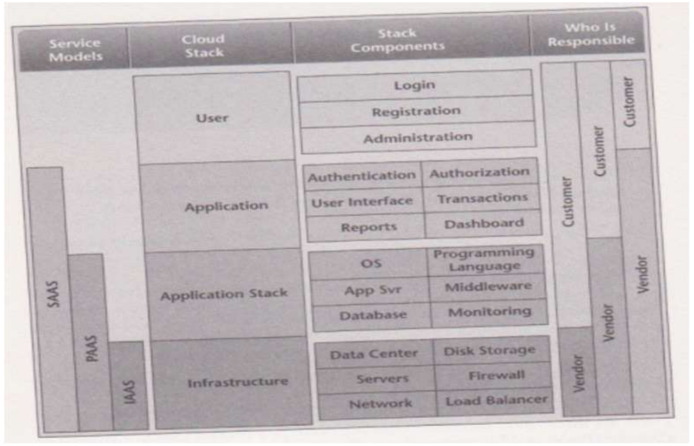
Cloud services

- Infrastructure as a Services (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)
- Cloud types:
  - Private
  - Public
  - Hybrid
  - Community

45

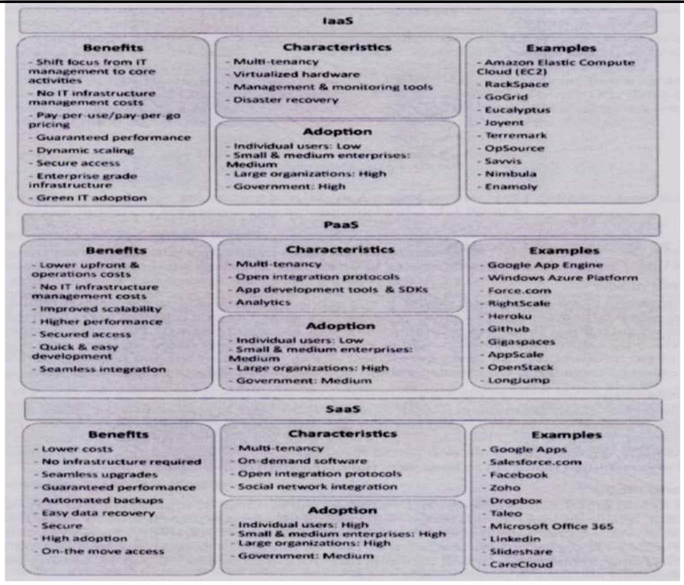
45

The Cloud Stack



46

46



IaaS

Benefits
- Shift focus from IT management to core activities
- No IT infrastructure management costs
- Pay-per-use/pay-per-go pricing
- Guaranteed performance
- Dynamic scaling
- Secure access
- Enterprise grade infrastructure
- Green IT adoption

Characteristics
- Multi-tenancy
- Virtualized hardware
- Management & monitoring tools
- Disaster recovery

Adoption
- Individual users: Low
- Small & medium enterprises: Medium
- Large organizations: High
- Government: High

Examples
- Amazon Elastic Compute Cloud (EC2)
- RackSpace
- GoGrid
- Eucalyptus
- Joyent
- Terremark
- OpSource
- Savvis
- Nimbula
- Enamoly

PaaS

Benefits
- Lower upfront & operations costs
- No IT infrastructure management costs
- Improved scalability
- Higher performance
- Secured access
- Quick & easy development
- Seamless integration

Characteristics
- Multi-tenancy
- Open integration protocols
- App development tools & SDKs
- Analytics

Adoption
- Individual users: Low
- Small & medium enterprises: Medium
- Large organizations: High
- Government: Medium

Examples
- Google App Engine
- Windows Azure Platform
- Force.com
- RightScale
- Heroku
- Github
- Gigaspaces
- AppScale
- OpenStack
- LongJump

SaaS

Benefits
- Lower costs
- No infrastructure required
- Seamless upgrades
- Guaranteed performance
- Automated backups
- Easy data recovery
- Secure
- High adoption
- On-the move access

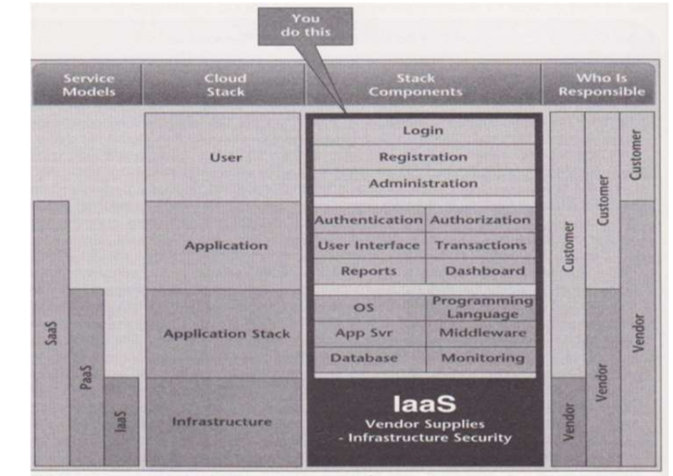
Characteristics
- Multi-tenancy
- On-demand software
- Open integration protocols
- Social network integration

Adoption
- Individual users: High
- Small & medium enterprises: High
- Large organizations: High
- Government: Medium

Examples
- Google Apps
- Salesforce.com
- Facebook
- Zoho
- Dropbox
- Taleo
- Microsoft Office 365
- Linkedin
- Slideshare
- CareCloud

47

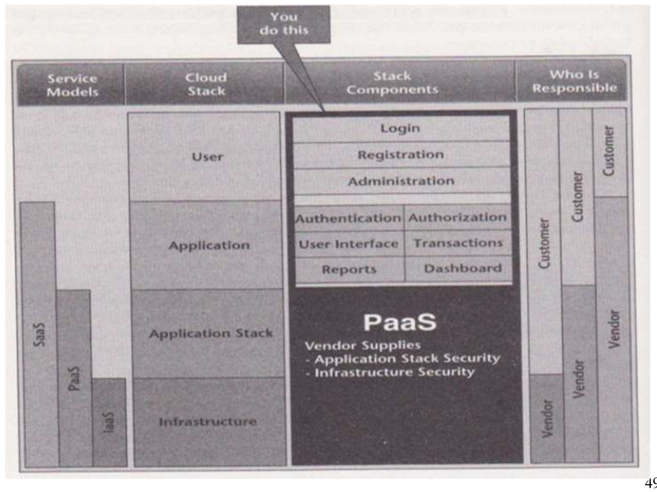
IaaS – Your Responsibility



48

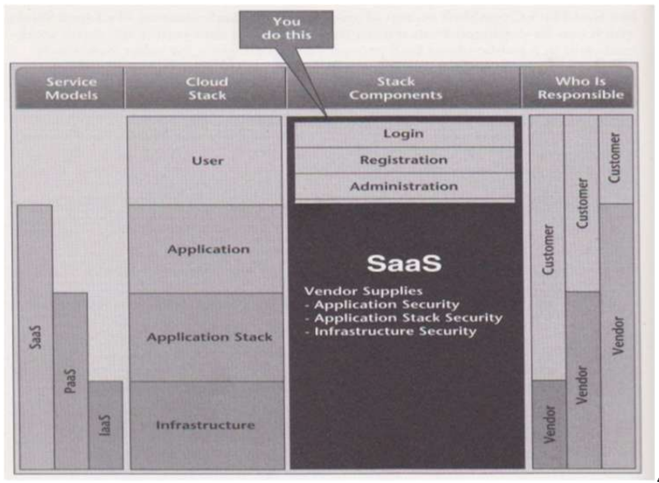
48

PaaS – Your Responsibility



49

SaaS – Your Responsibility



50

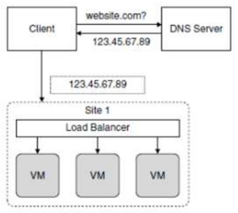
Features of the Cloud - Virtualization

- The fundamental enabler of the cloud is virtualization over hundreds of thousands of hosts accessible over the Internet.
  - In cloud computing, a virtual machine (VM) is an emulation of a physical machine. A VM image is a file that contains a bootable operating system and some software
  - When using IaaS, a consumer acquires a VM from a VM image by using an application programming interface (API) provided by the cloud provider for that purpose.
  - The process of creating a VM image is called *baking* the image. A *heavily* baked image contains all of the software required to run an application and a *lightly* baked image contains only a portion of the software required, such as an operating system and a middleware container.

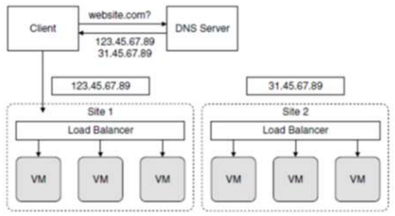
51

IP and Domain Name System Management (DNS)

- When a VM is created, it is assigned an IP address. IP addresses are the means by which messages are routed to any computer on the Internet.
- When you enter a URL into your browser, it sends that URL to its known DNS server which, in association with a larger network of DNS servers, resolves that URL into an IP address.



DNS returning an IP address



DNS returning two addresses for a single URL

52

Persistence of IP Addresses with Respect to VMs

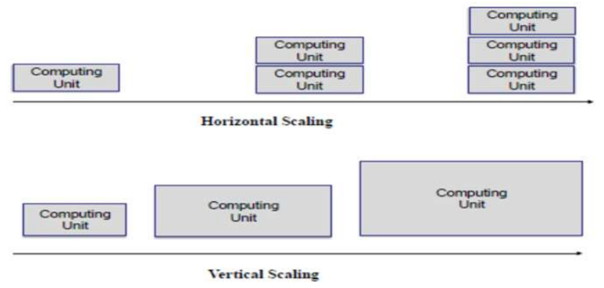
- The IP address assigned to a virtual machine on its creation persists as long as that VM is active.
- A VM becomes inactive when it is terminated, paused, or stopped.
- In these cases, the IP address is returned to the cloud provider's pool for reassignment.
- A consequence of IP reassignment is: If one VM within your application sends a message to another VM within your application it must verify that the IP address of the recipient VM is still current.
  - Consider the following sequence where your application contains at least VMA and VMB.
    1. VM_B receives a message from VM_A.
    2. VM_A fails.
    3. The cloud provider reassigns the IP address of VM_A.
    4. VM_B responds to the originating IP address.
    5. The message is delivered to a VM that is not a portion of your application.

53

53

Scaling I – Horizontal vs Vertical

- VMs are common unit of resource which can be configured.
- **Horizontal scaling** - Public providers such as Amazon offer a variety of VM templates in terms of the CPU, RAM or storage to be added/removed during execution of the application.
- **Vertical scaling** - On the other hand, recent hardware and software advancements make fine-grained controls on resources possible. This enable the on-the-fly resizing of an active VM to adjust the amount of allocated resources to the workload of the VM.

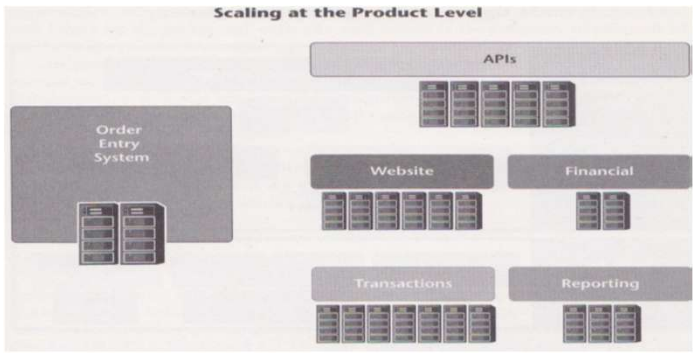


54

54

Scaling in the Cloud II

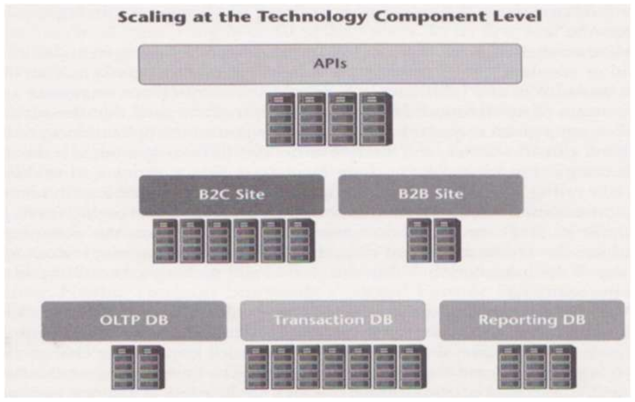
Horizontal scaling is done by adding additional infrastructure that runs in conjunction with the existing infrastructure. Common horizontal scaling methods are to add nodes by server farm type (figure below and next slide).



55

55

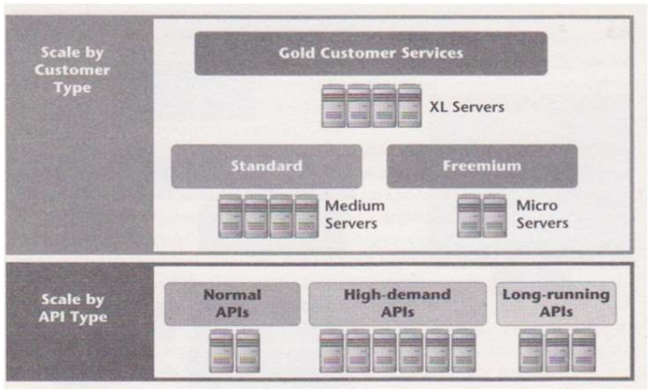
Scaling in the Cloud III



56

56

Scale by Customer & by API IV

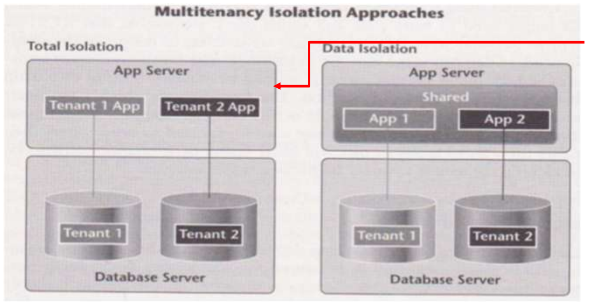


57

57

Multitenant or Single Tenant

- The tenancy of a system is generally determined by the data characteristics.
- Single tenant – means that only one tenant is supported per each group of servers
- Multitenancy – means multiple organizations or customers (tenants) share a group of servers.



Total Isolation – An example of Single tenant

58

58

Total Isolation

Advantages	**Disadvantages**
• Provides independence	Most expensive
• Privacy	Minimal reuse
• Highest scalability	Highest complexity

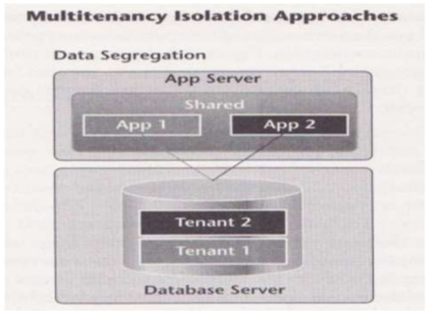
Data Isolation strategy

- Multitenant approach at the application layer
- The database layer is single tenant making it hybrid between multitenancy and single tenant.
- This model (i.e. total isolation) provides independence and privacy while reducing the cost of complexity.

59

59

Data Segregation



Advantages
- Most cost effective
- Least complex
- Highest reuse

Disadvantages
- Lack of independence
- Lowest performance
- Lowest scalability

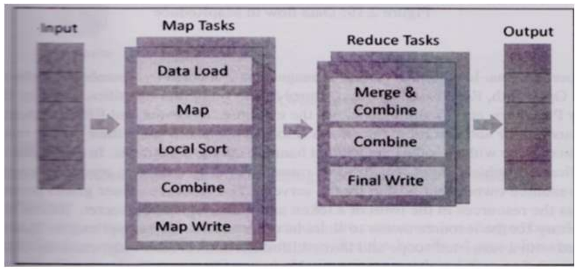
- The data segregation strategy separates the tenants into different database schemes, but they do share the same servers.
- In this model, all layers are shared for all tenants
- This model is effective because of reuse.

60

60

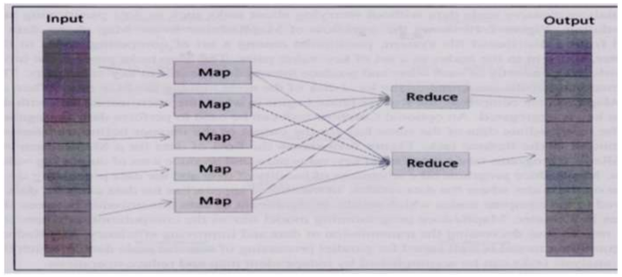
MapReduce I

- MapReduce is a parallel data processing model for processing and analysis of massive scale data.
- Two phases: Map and Reduce



- In map phase – data is read from a distributed file system.
- Figure above shows the workflow of map reduce.

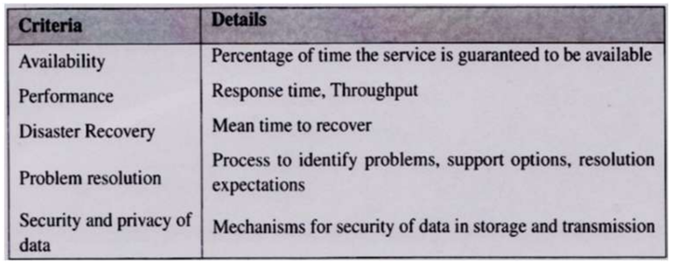
MapReduce I



- The Map tasks process the input independently and produce intermediate results as key-value pairs.
- An optional combine task is used for data segregation before sending the output to Reduce task.
- Figure above shows the flow of data for a MapReduce.

Service Level Agreements (SLA)

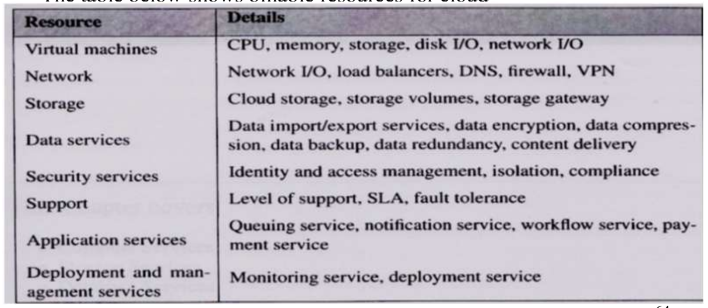
- SLA specifies the level of service that is formally defined as a part of the service contract with the cloud service provider.
- This is specified in form of minimum level of service guaranteed and a target level. Below is a list of common criteria for cloud SLAs



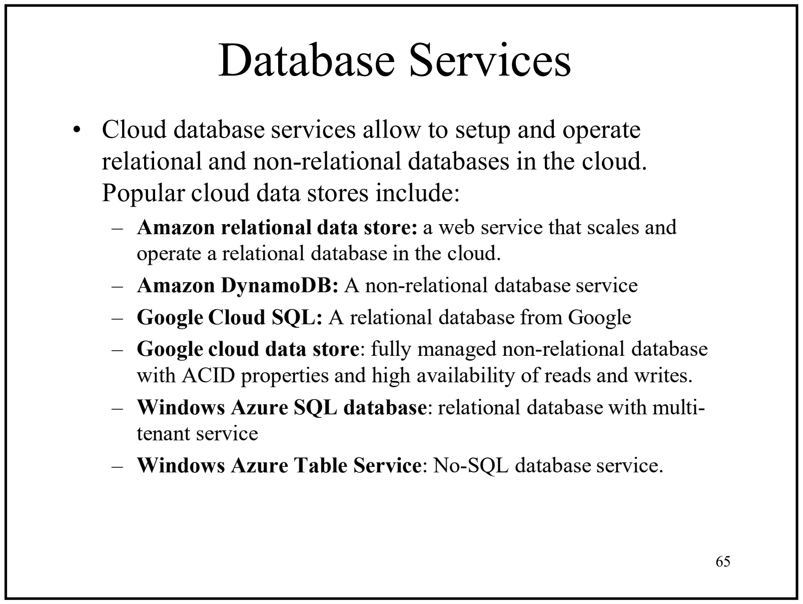
Criteria	Details
Availability	Percentage of time the service is guaranteed to be available
Performance	Response time, Throughput
Disaster Recovery	Mean time to recover
Problem resolution	Process to identify problems, support options, resolution expectations
Security and privacy of data	Mechanisms for security of data in storage and transmission

Cloud Billing

- Cloud service providers offer a number of billing models:
  - Elastic pricing – pay-as-you-use pricing model
  - Fixed pricing – customers are charged a fixed amount per month for the cloud resources
  - Spot pricing – offers variable pricing for cloud resources which is driven by market demand.
- The table below shows billable resources for cloud



Resource	Details
Virtual machines	CPU, memory, storage, disk I/O, network I/O
Network	Network I/O, load balancers, DNS, firewall, VPN
Storage	Cloud storage, storage volumes, storage gateway
Data services	Data import/export services, data encryption, data compression, data backup, data redundancy, content delivery
Security services	Identity and access management, isolation, compliance
Support	Level of support, SLA, fault tolerance
Application services	Queuing service, notification service, workflow service, payment service
Deployment and management services	Monitoring service, deployment service

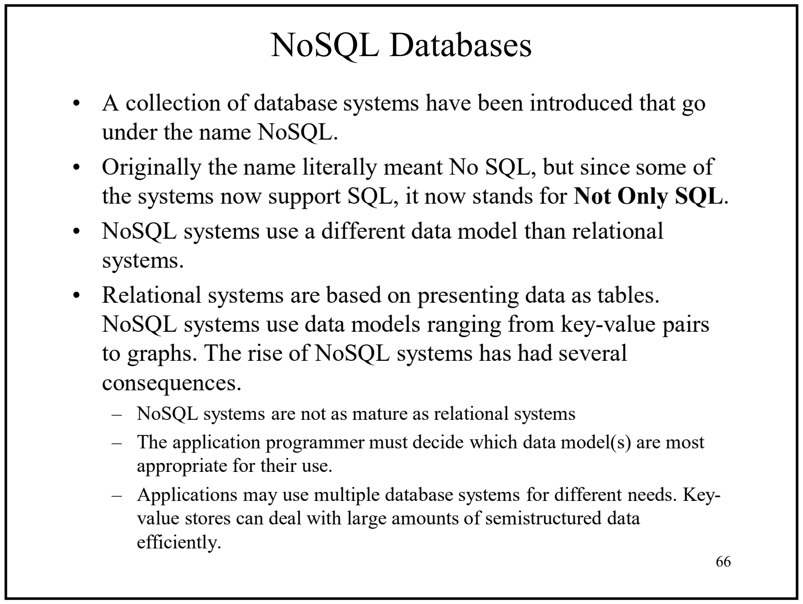


Database Services

- Cloud database services allow to setup and operate relational and non-relational databases in the cloud. Popular cloud data stores include:
  - **Amazon relational data store:** a web service that scales and operate a relational database in the cloud.
  - **Amazon DynamoDB:** A non-relational database service
  - **Google Cloud SQL:** A relational database from Google
  - **Google cloud data store**: fully managed non-relational database with ACID properties and high availability of reads and writes.
  - **Windows Azure SQL database**: relational database with multi-tenant service
  - **Windows Azure Table Service**: No-SQL database service.

65

65

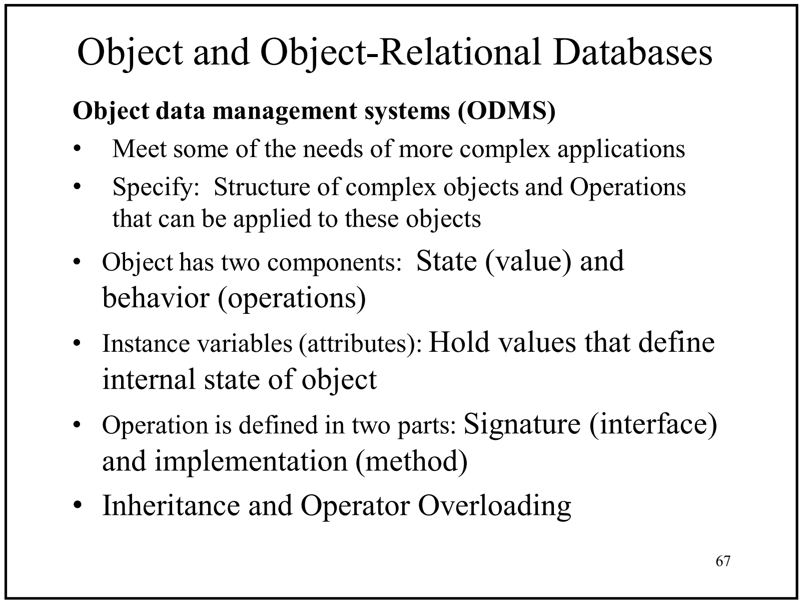


NoSQL Databases

- A collection of database systems have been introduced that go under the name NoSQL.
- Originally the name literally meant No SQL, but since some of the systems now support SQL, it now stands for **Not Only SQL**.
- NoSQL systems use a different data model than relational systems.
- Relational systems are based on presenting data as tables. NoSQL systems use data models ranging from key-value pairs to graphs. The rise of NoSQL systems has had several consequences.
  - NoSQL systems are not as mature as relational systems
  - The application programmer must decide which data model(s) are most appropriate for their use.
  - Applications may use multiple database systems for different needs. Key-value stores can deal with large amounts of semistructured data efficiently.

66

66



Object and Object-Relational Databases

Object data management systems (ODMS)

- Meet some of the needs of more complex applications
- Specify: Structure of complex objects and Operations that can be applied to these objects
- Object has two components: State (value) and behavior (operations)
- Instance variables (attributes): Hold values that define internal state of object
- Operation is defined in two parts: Signature (interface) and implementation (method)
- Inheritance and Operator Overloading

67

67