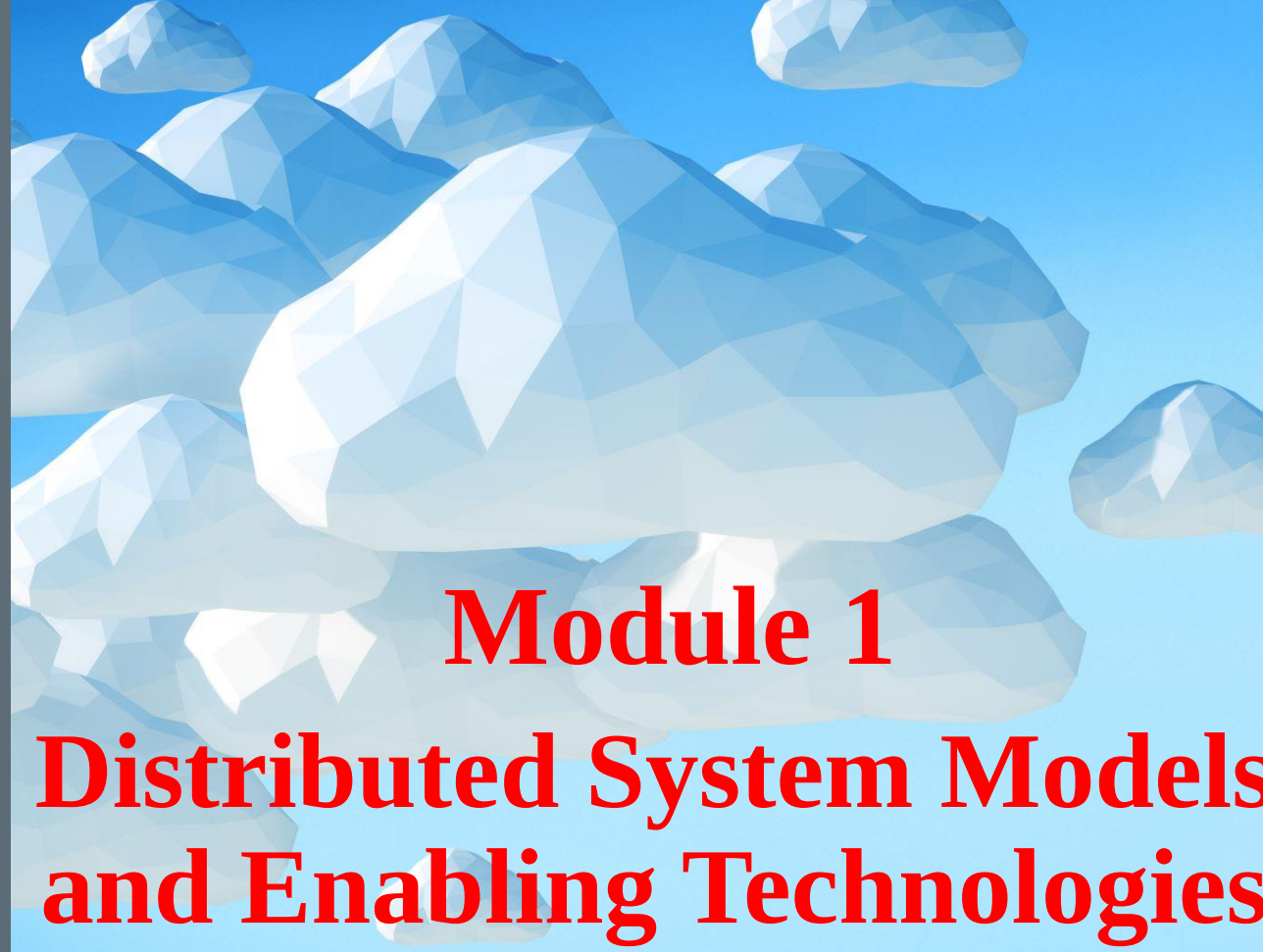
Cloud Computing and Security

BIS613D

Department of ISE



Module 1

Distributed System Models and Enabling Technologies

CELEBRATING 25 YEARS

CMRIT

CMR INSTITUTE OF TECHNOLOGY, BENGALURU.

ACCREDITED WITH A++ GRADE BY NAAC

Service-Oriented Architecture (SOA)

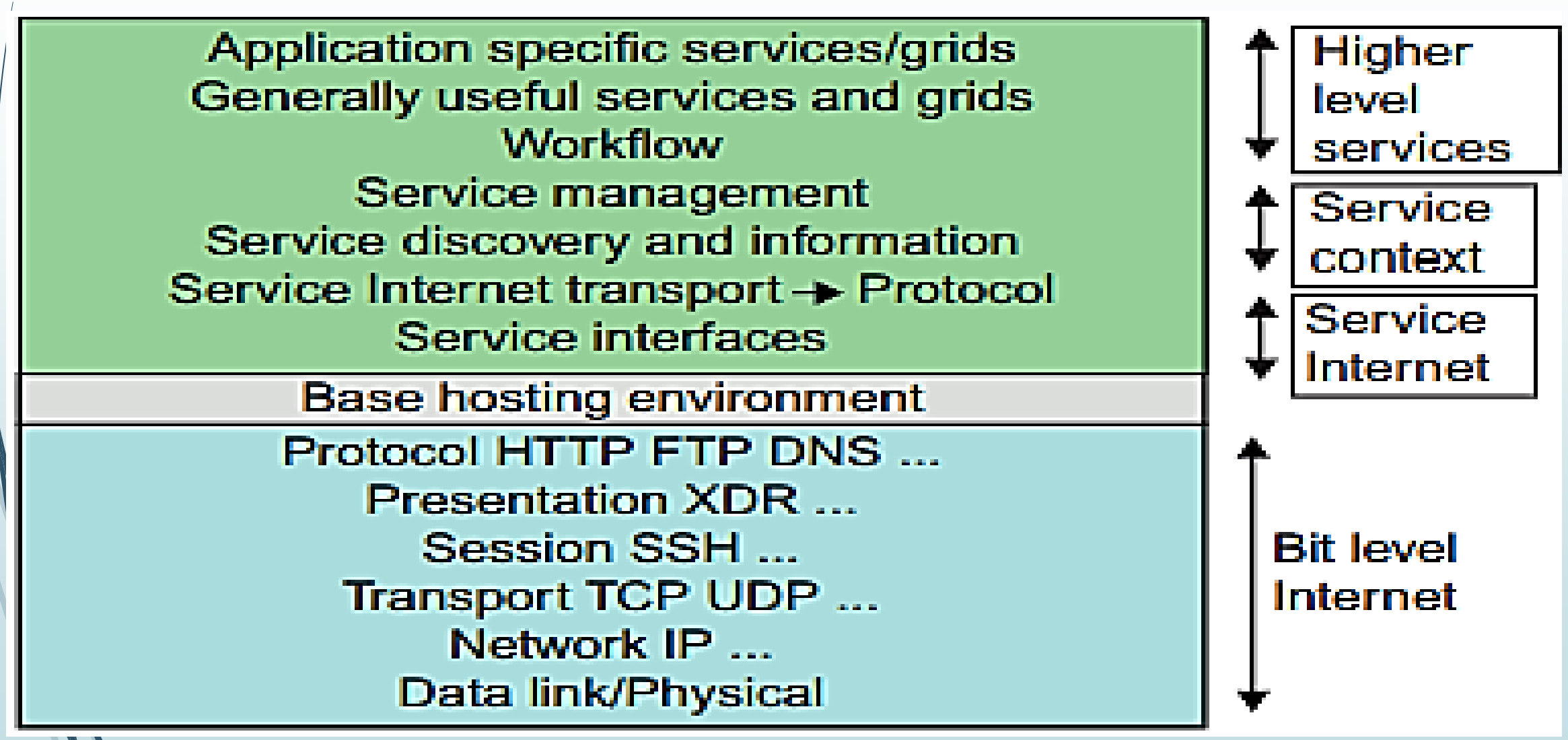
Term	Full Form	Description	Key Features
CORBA	Common Object Request Broker Architecture	A middleware standard that enables communication between objects in a distributed, heterogeneous environment.	- Supports distributed objects- Uses Object Request Brokers (ORBs)- Language and platform agnostic- Uses Interface Definition Language (IDL)
SOAP	Simple Object Access Protocol	A protocol for exchanging structured information in the implementation of web services.	- XML-based messaging protocol- Supports platform and language independence- Designed for request-response communication- Can work over HTTP, SMTP, etc.
REST	Representational State Transfer	An architectural style for building web services that use stateless communication and standard HTTP methods.	- Simple and lightweight- Stateless communication- Uses standard HTTP methods (GET, POST, PUT, DELETE)- Focuses on resources and their representations (URLs)

Service-Oriented Architecture (SOA)

- In grids/web services, Java, and CORBA, an entity is a service, a Java object, and a CORBA distributed object in a variety of languages.
- These architectures build on the traditional seven OSI layers that provide the base networking abstractions.
- Figure1.20 shows the layered architecture for distributed entities used in web services and grid systems.

Service-Oriented Architecture (SOA)

FIGURE 1.20 Layered Architecture for Web services and the grids.



Layered Architecture for Web Services and Grids

- The entity interfaces correspond to the Web Services Description Language (WSDL), Java method, and CORBA interface definition language (IDL) specifications in these example distributed systems.
- These interfaces are linked with customized, high-level communication systems: SOAP, RMI, and IIOP in the three examples
- In the case of fault tolerance, the features in the Web Services Reliable Messaging (WSRM) framework mimic the OSI layer capability modified to match the different abstractions at the entity levels.

Web Services and Tools

Point	Description
Web Services vs REST	Web services are more rigid, using SOAP for communication, while REST focuses on simplicity and flexibility.
Web Services (SOAP)	Web services aim to fully specify the service and environment, using SOAP for distributed communication.
REST Architecture	REST is simpler, using minimal headers and an opaque message body, making it suitable for rapid technology environments.
XML in REST vs SOAP	REST can use XML schemas but not those used in SOAP, with "XML over HTTP" being a common design choice.
Composite Applications and RPC	In CORBA and Java, distributed entities are linked via RPC, with CORBA using object interfaces similar to C++ for linking.

The Evolution of SOA

1. SOA and Distributed Systems: SOA is applicable for building complex systems like grids, clouds, interclouds (grids of clouds), and systems of systems, supporting interoperability across diverse services.
2. Sensor Services (SS): Sensors (e.g., ZigBee, Bluetooth, WiFi, GPS, wireless phones) provide raw data through sensor services, which are crucial in data collection for various systems.

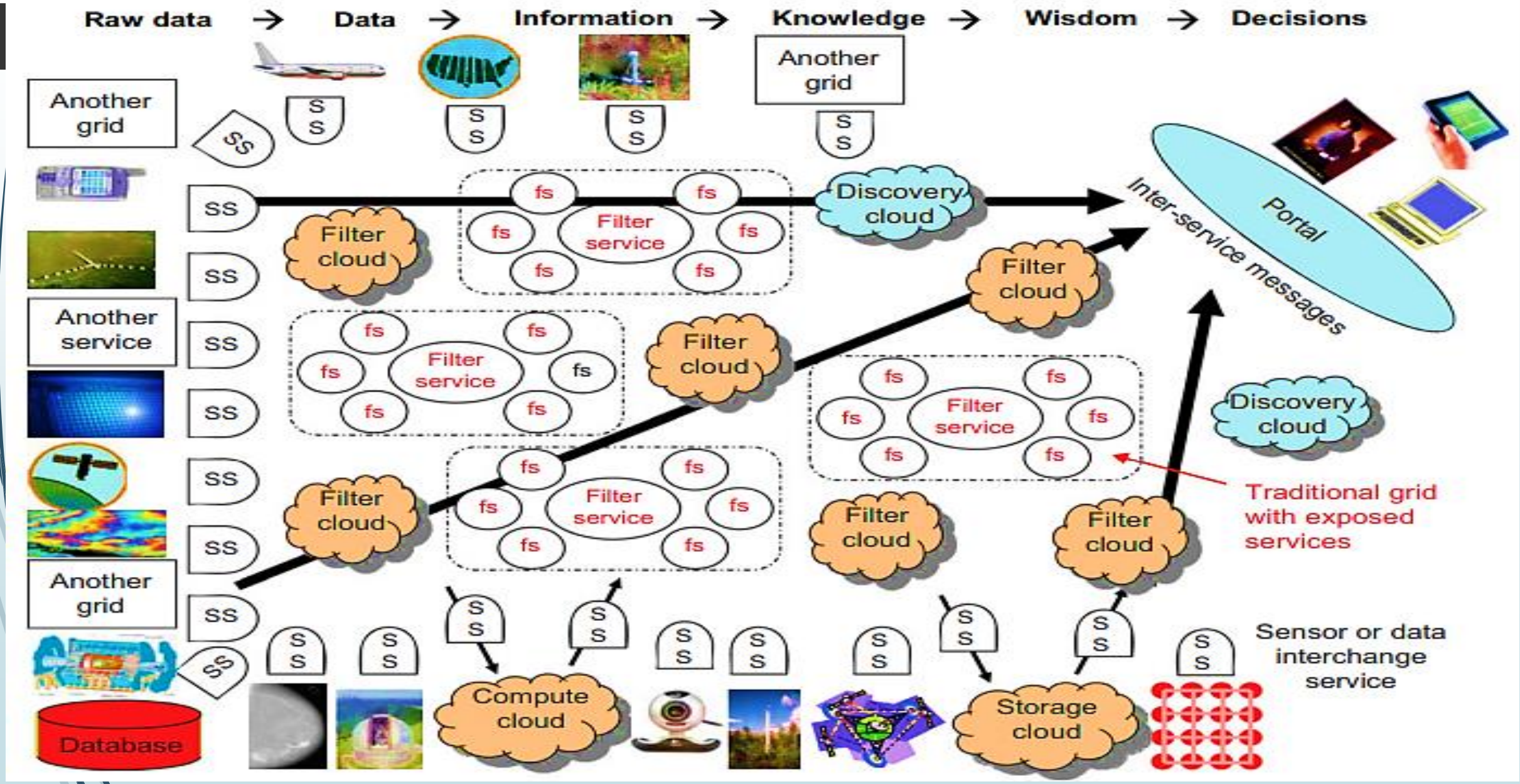
3. Data Collection and Interaction: Sensor services (SS) interact with various computing entities like small or large computers, grids, and cloud services to manage and process collected data.

The Evolution of SOA

4. Clouds and Services: The collected data is processed across different types of clouds such as compute, storage, filter, and discovery clouds, each playing a specific role in managing data and services.

5. Filter Services (FS): Filter services (FS) are used to process and clean raw data, ensuring that only relevant data is passed on to respond to specific requests from web services, grids, or the cloud.

Figure 1.21, service-oriented architecture (SOA) has evolved over the years.



Grids versus Clouds

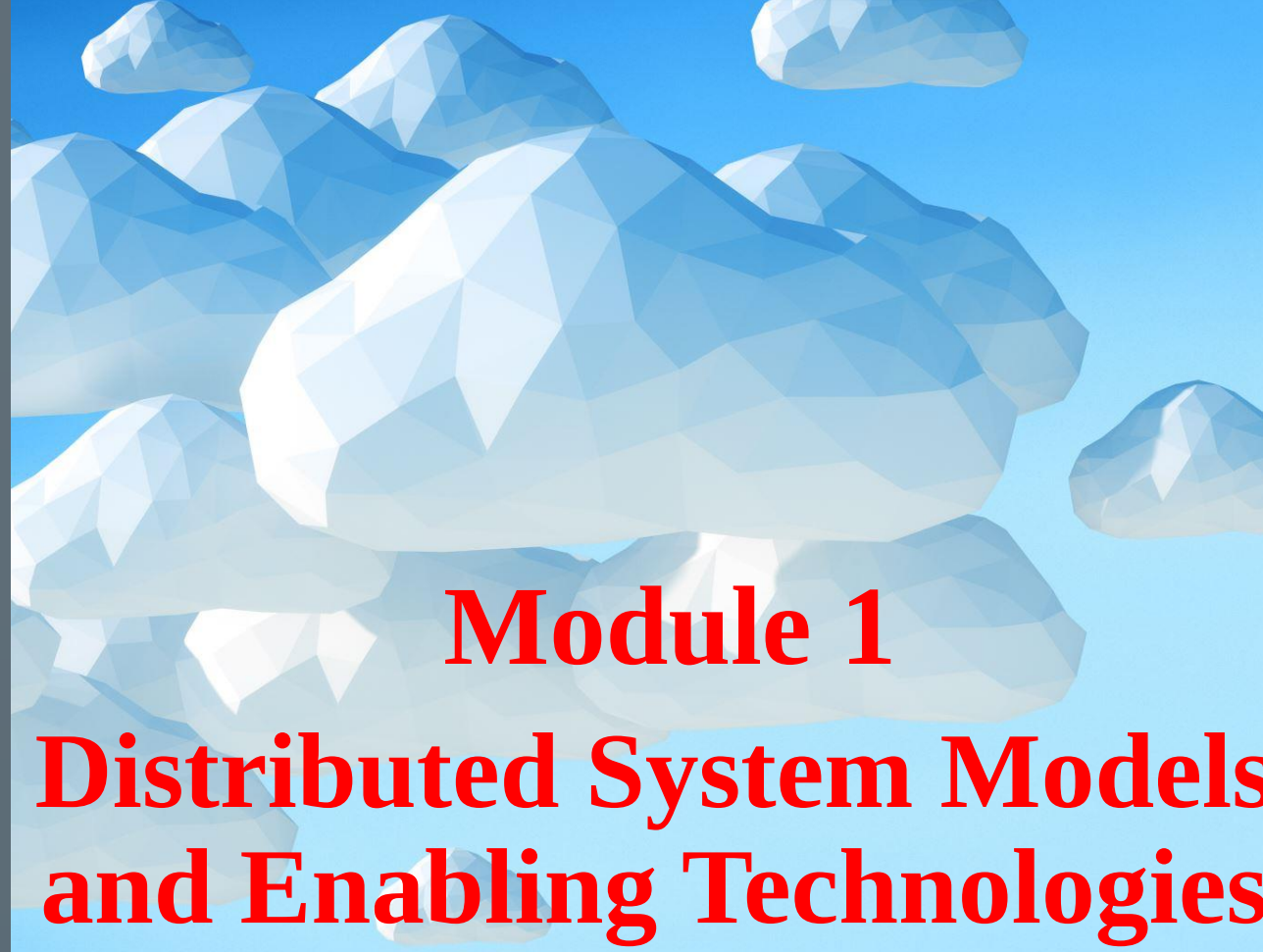
Aspect	Grids	Clouds
Resource Type	Applies static resources for fixed computing needs.	Emphasizes elastic, scalable resources based on demand.
Dynamic Resource Allocation	Limited dynamic allocation, more static in nature.	Dynamic resource allocation enabled by virtualization and autonomic computing.
System Composition	A collection of fixed resources, often managed statically.	Composed of scalable resources with flexible allocation capabilities.
System Flexibility	Less flexible in terms of resource management and scaling.	Highly flexible, capable of scaling resources up and down based on demand.
Hybrid Models	A grid can be built from multiple clouds, allowing negotiated resource allocation.	Can integrate with grids (cloud of grids, grid of clouds, inter-clouds) for better resource management.

THANK YOU

Cloud Computing and Security

BIS613D

Department of ISE



Module 1

Distributed System Models and Enabling Technologies

CELEBRATING 25 YEARS

CMRIT

CMR INSTITUTE OF TECHNOLOGY, BENGALURU.

ACCREDITED WITH A++ GRADE BY NAAC

SOFTWARE ENVIRONMENTS FOR DISTRIBUTED SYSTEMS AND CLOUDS

- **Trends toward Distributed Operating Systems**
- **Parallel and Distributed Programming Models**

Trends toward Distributed Operating Systems

- Distributed Operating Systems
- Amoeba versus DCE
- MOSIX2 for Linux Clusters
- Transparency in Programming Environments

Distributed Operating Systems

- Three approaches for distributing resource management functions in a distributed computer system.
- ➢ First approach is to build a network OS over a large number of heterogeneous OS platforms.
- ➢ Second approach is to develop middleware to offer a limited degree of resource sharing.
- ➢ Third approach is to develop a truly distributed OS to achieve higher use or system transparency.

Distributed Operating Systems

Table 1.6 Feature Comparison of Three Distributed Operating Systems

Distributed OS Functionality	AMOEBA Developed at Vrije University [46]	DCE as OSF/1 by Open Software Foundation [7]	MOSIX for Linux Clusters at Hebrew University [3]
History and Current System Status	Written in C and tested in the European community; version 5.2 released in 1995	Built as a user extension on top of UNIX, VMS, Windows, OS/2, etc.	Developed since 1977, now called MOSIX2 used in HPC Linux and GPU clusters
Distributed OS Architecture	Microkernel-based and location-transparent, uses many servers to handle files, directory, replication, run, boot, and TCP/IP services	Middleware OS providing a platform for running distributed applications; The system supports RPC, security, and threads	A distributed OS with resource discovery, process migration, runtime support, load balancing, flood control, configuration, etc.
OS Kernel, Middleware, and Virtualization Support	A special microkernel that handles low-level process, memory, I/O, and communication functions	DCE packages handle file,time, directory, security services, RPC, and authentication at middleware or user space	MOSIX2 runs with Linux 2.6; extensions for use in multiple clusters and clouds with provisioned VMs
Communication Mechanisms	Uses a network-layer FLIP protocol and RPC to implement point-to-point and group communication	RPC supports authenticated communication and other security services in user programs	Using PVM, MPI in collective communications, priority process control, and queuing services

Amoeba versus DCE

1.Research Prototypes in Distributed Computing: Systems like Amoeba, DCE, and MOSIX2 are research prototypes developed for distributed computing, primarily used in academia.

2.Need for Web-Based Operating Systems: to support the virtualization of resources in distributed environments, an area still under active research.

3.Distributed Resource Management: A distributed operating system should balance resource management by distributing its functionalities across available servers.

4.Microkernel and OS Extensions: The design of such distributed OSs could either adopt a lightweight microkernel approach, like the Amoeba, or extend existing systems like DCE or UNIX for effective distribution of OS services.

5.Focus on User Autonomy: The trend is to simplify user responsibilities, freeing them from most resource management tasks

MOSIX2 for Linux Clusters

1.Distributed OS in Linux: MOSIX2 is a distributed operating system that operates within a Linux environment, using a virtualization layer.

2.Partial Single-System Image: enabling them to interact as if they were on a single system.

3.Support for Sequential and Parallel Applications: allowing flexibility in processing tasks.

4.Resource Discovery and Process Migration: The system discovers resources and can migrate software processes between different Linux nodes, optimizing resource usage.

5.Cluster and Grid Management: MOSIX2 can manage a Linux cluster or a grid made up of multiple clusters.

MOSIX2 for Linux Clusters

1.Distributed OS in Linux: MOSIX2 is a distributed operating system that operates within a Linux environment, using a virtualization layer.

2.Partial Single-System Image: enabling them to interact as if they were on a single system.

3.Support for Sequential and Parallel Applications: allowing flexibility in processing tasks.

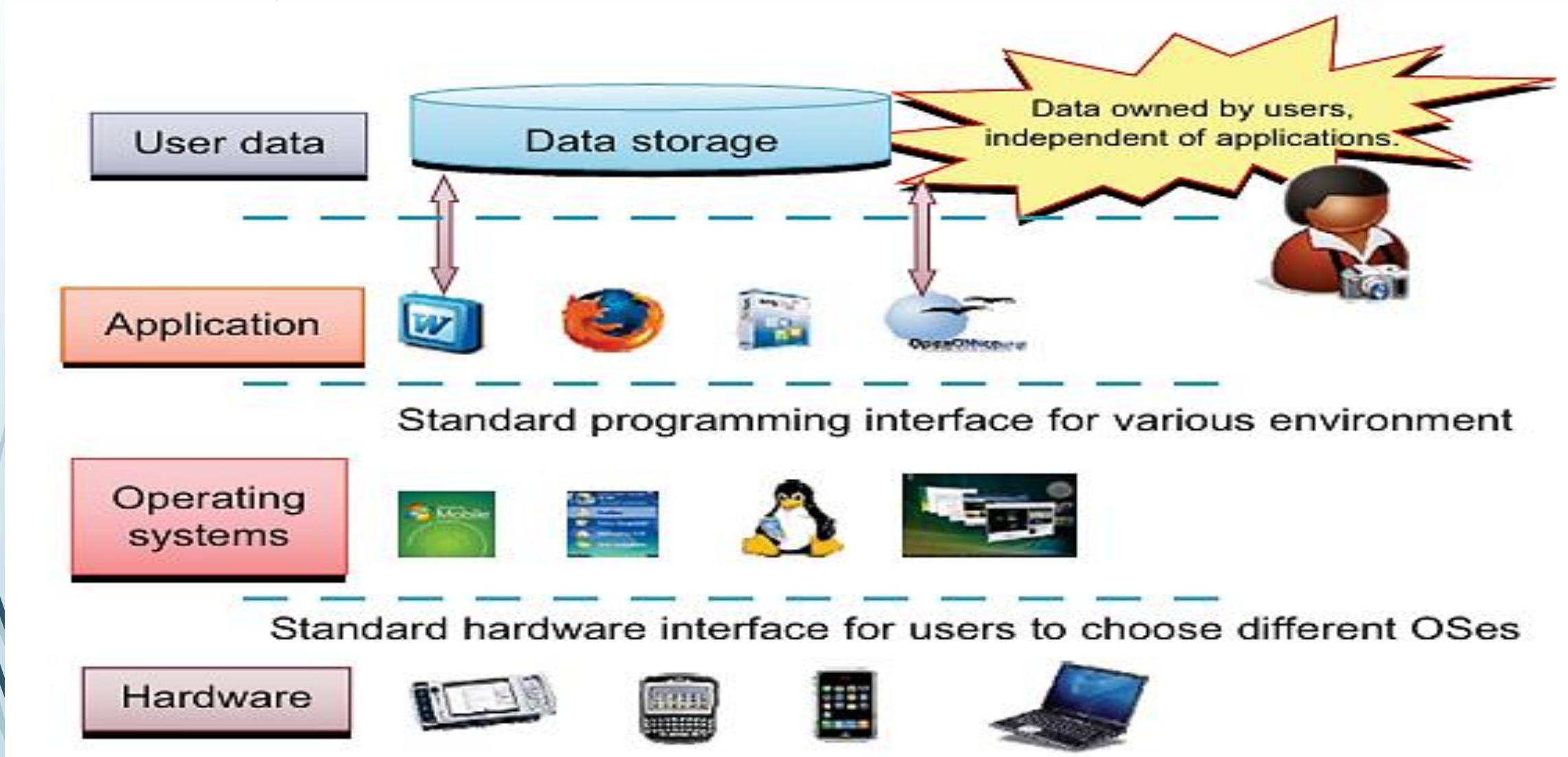
4.Resource Discovery and Process Migration: The system discovers resources and can migrate software processes between different Linux nodes, optimizing resource usage.

5.Cluster and Grid Management: MOSIX2 can manage a Linux cluster or a grid made up of multiple clusters.

Transparency in Programming Environments

- Fig.1.22 shows the concept of a transparent computing infrastructure for future computing platforms.
- The user data, applications, OS, and hardware are separated into four levels.
- Data is owned by users, independent of the applications.
- The OS provides clear interfaces, standard programming interfaces, or system calls to application programmers.
- To separate user data from specific application programs, users can enable cloud applications as SaaS.

FIG 1.22 A transparent computing environment that separates the user data, application, OS, and hardware in time and space– an ideal model for cloud computing.



Parallel and Distributed Programming Models

- Four programming models for distributed computing with expected scalable performance and application flexibility.
- Table 1.7 summarizes three of these models, along with some software tool sets developed in recent years.
- MPI is the most popular programming model for message-passing systems.
- Google’s MapReduce and BigTable are for effective use of resources from Internet clouds and data centers.

Parallel and Distributed Programming Models

Table 1.7 Parallel and Distributed Programming Models and Tool Sets

Model	Description	Features
MPI	A library of subprograms that can be called from C or FORTRAN to write parallel programs running on distributed computer systems [6,28,42]	Specify synchronous or asynchronous point-to-point and collective communication commands and I/O operations in user programs for message-passing execution
MapReduce	A web programming model for scalable data processing on large clusters over large data sets, or in web search operations [16]	*Map* function generates a set of intermediate key/value pairs; *Reduce* function merges all intermediate values with the same key
Hadoop	A software library to write and run large user applications on vast data sets in business applications (http://hadoop.apache.org/core)	A scalable, economical, efficient, and reliable tool for providing users with easy access of commercial clusters

Parallel and Distributed Programming Models

- Message-Passing Interface (MPI)
- MapReduce
- Hadoop Library
- Open Grid Services Architecture (OGSA)
- Globus Toolkits and Extensions

Message-Passing Interface (MPI)

- Primary programming standard used to develop parallel and concurrent programs to run on a distributed system.
- MPI is essentially a library of subprograms that can be called from C or FORTRAN to write parallel programs running on a distributed system.
- The idea is to embody clusters, grid systems, and P2P systems with upgraded web services and utility computing applications.

MapReduce

- A web programming model for scalable data processing on large clusters over large data sets.
- The model is applied mainly in web-scale search and cloud computing applications.
- The user specifies a Map function to generate a set of intermediate key/value pairs.
- Then the user applies a Reduce function to merge all intermediate values with the same intermediate key.
- MapReduce is highly scalable to explore high degrees of parallelism at different job levels.
- A typical MapReduce computation process can handle terabytes of data on tens of thousands or more client machines.