IoT Enabling Technologies: Wireless Sensor Networks

A Wireless Sensor Network (WSN) is a collection of devices which communicate through wireless channels. A WSN consists of distributed devices with sensors which are used to monitor the environmental and physical conditions.

A WSN consists of a number of end nodes, routers and coordinators. End nodes can also act as routers. A coordinator collects data from all the nodes and is connected to Internet.

Examples of WSNs used in IoT systems:

Weather monitoring systems
Indoor air quality monitoring systems
Soil moisture monitoring systems
Surveillance systems
Smart grids
Structural health monitoring systems

IoT Enabling Technologies: Cloud Computing.

Cloud computing is a computing model in which applications and services are delivered over Internet. The resources provisioned by cloud can be compute, networking or storage. Cloud allows the users to access resources based on utility model. The characteristics of cloud computing are:

On demand: The resources in the cloud are available based on the traffic. If the incoming traffic increases, the cloud resources scale up accordingly and when the traffic decreases, the cloud resources scale down accordingly.

Autonomic: The resource provisioning in the cloud happens with very less to no human intervention. The resources scale up and scale down automatically.

Scalable: The cloud resources scale up and scale down based on the demand or traffic. This property of cloud is also known as elasticity.

Pay-per-use: On contrary to traditional billing, the cloud resources are billed on pay-per-use basis. You have to pay only for the resources and time for which you are using those resources.

Ubiquitous: You can access the cloud resources from anywhere in the world from any device. All that is needed is Internet. Using Internet you can access your files, databases and other resources in the cloud from anywhere.

Cloud computing offers three basic service models using which users can subscribe to cloud resources. These service models are:

Infrastructure-As-A-Service (IAAS)
Platform-As-A-Service (PAAS)
Software-As-A-Service (SAAS)

Deployment models are

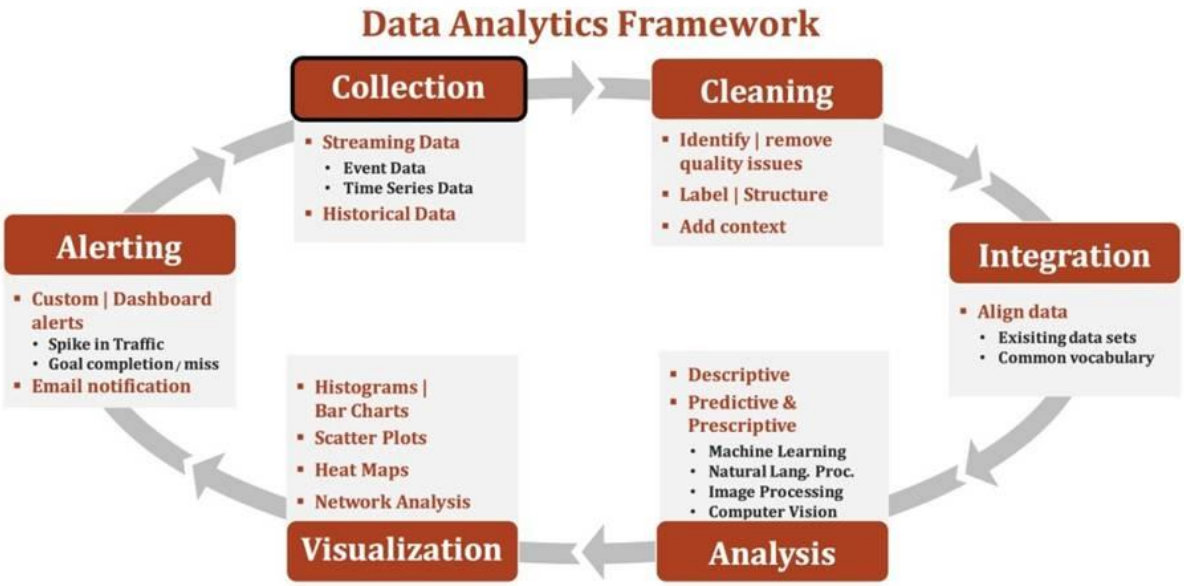
Public cloud
Private cloud
Community cloud
Hybrid cloud

IoT Enabling Technologies: BigData Analytics

BigData is a collection of data coming from various types of sources. The data is often huge which cannot be handled by the traditional databases and data warehouses. BigData is often characterized by six Vs. They are:

Volume: Refers to the huge volume of data aggregated from various sources.
Variety: Refers to different types of data. Data can be structured, semi-structured or unstructured.
Velocity: Refers to the speed at which the data is generated. Now-a-days the amount of data available on the Internet per minute is several peta bytes or even more.
Veracity: Refers to the degree to which the data can be trusted. If the data collected is incorrect or has manipulated or wrong values, the analysis of such data is useless.
Value: Refers to the business value of the collected. Even though we have huge amount of data, but it is not useful for gaining profits in the business, such data is useless.
Variability: Refers to the ways in which the big data can be used and formatted.

The data analytics framework consists of six steps namely: collection, cleaning, integration, analysis, visualization and alerting. These six steps can be summarized as shown in the below figure.



IoT Enabling Technologies: Communication Protocols

Communications protocols form the backbone for IoT systems. They allow devices to communicate with each other. Protocols define the data exchange formats, data encoding and addressing schemes for devices. Protocols also provide flow control, error control, and other functions.

IoT Enabling Technologies: Embedded Systems

Embedded system can be imagined as computing hardware with software embedded in it. An embedded system can be an independent system or it can be a part of another larger system.

An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. The key components include microcontroller/micrprocessor, memory, networking units, I/O, and storage. It runs Real-Time Operating Systems (RTOS).

An embedded system has three components. They are:

Hardware
Software
Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per schedule by following a plan to control the latencies

The characteristics of an embedded system are:

Single-functioned
Tightly constrained
Reactive and Real time
Memory
Connected

IoT Levels – Deployment Templates

Developing an **IoT Level Template** system consists of the following components:

1. **Device:** These may be sensors or actuators capable of identifying, remote sensing, or monitoring.
2. **Resources:** These are software components on IoT devices for accessing and processing. storing software components or controlling actuators connected to the device. Resources also include software components that enable network access.

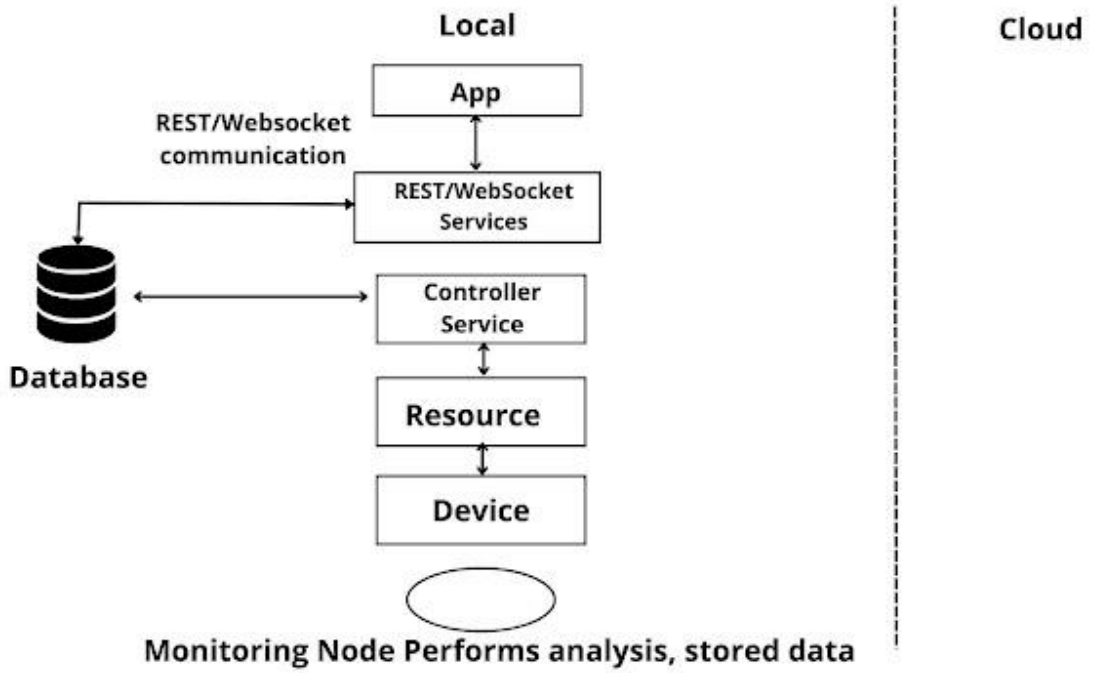
3. **Controller Service:** It is a service that runs on the device and interacts with web services. The controller service sends data from the device to the web service and receives commands from the application via web services for controlling the device.
4. **Database:** Stores data generated from the device
5. **Web Service:** It provides a link between IoT devices, applications, databases, and analysis components.
6. **Analysis Component:** It performs an analysis of the data generated by the lol device and generates results in a form which are easy for the user to understand.
7. **Application:** It provides a system for the user to view the system status and view product data. It also allows users to control and monitor various aspects of the IoT system.

IoT Levels

IoT level 1

IoT systems have a single device that performs sensing or actuation, stores a. analyses it, and hosts the application, IoT system-level-l is the best example for modeling low complexity and low-cost solution where the analysis requirement is not comprehensive and the data involved is not big.

Example: We can understand with the help of an eg. Let's look at the IoT device that monitors the lights in a house. The lights are controlled through switches. The database has maintained the status of each light and also REST services deployed locally allow retrieving and updating the state of each light and trigger the switches accordingly. For controlling the lights and applications, the application has an interface. The device is connected to the internet and hence the application can be accessed remotely as well.

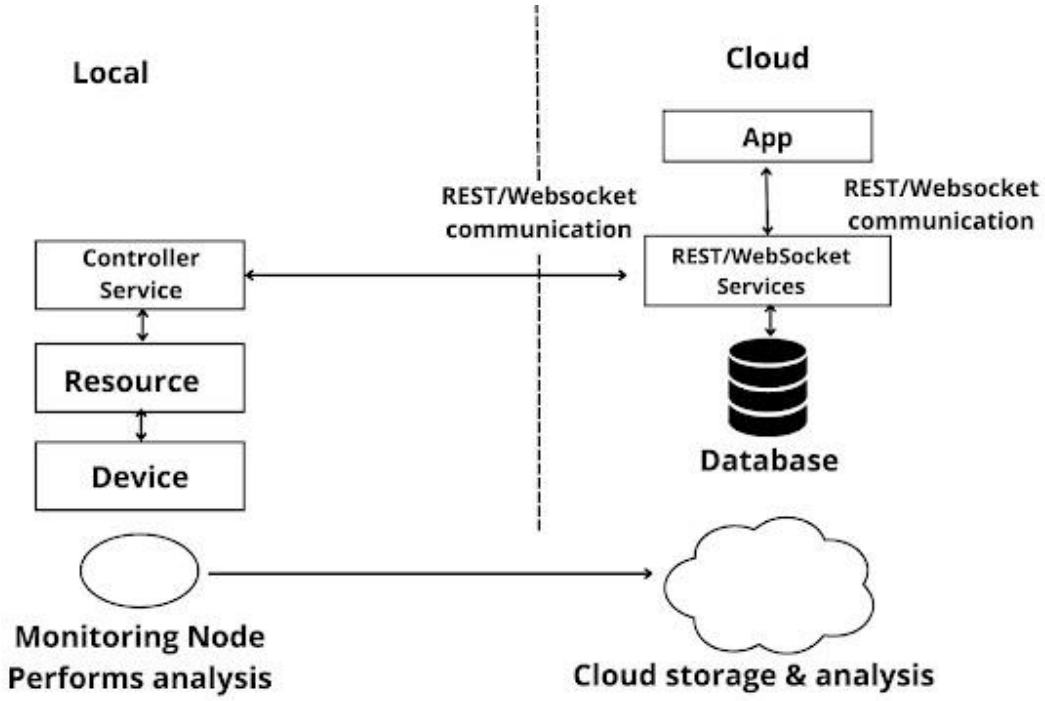


IoT Level-1

IoT level 2

A node performs sensing/actuation and local analysis. Data is stored in the cloud. this level is facilitated where the data involved is big and the primary analysis is not comprehensive.

Example: Cloud-based application is used for monitoring and controlling the IoT system A single node monitors the soil moisture in the field Which is sent to the database on the cloud using REST APIS. The controller service continuously monitors moisture levels.

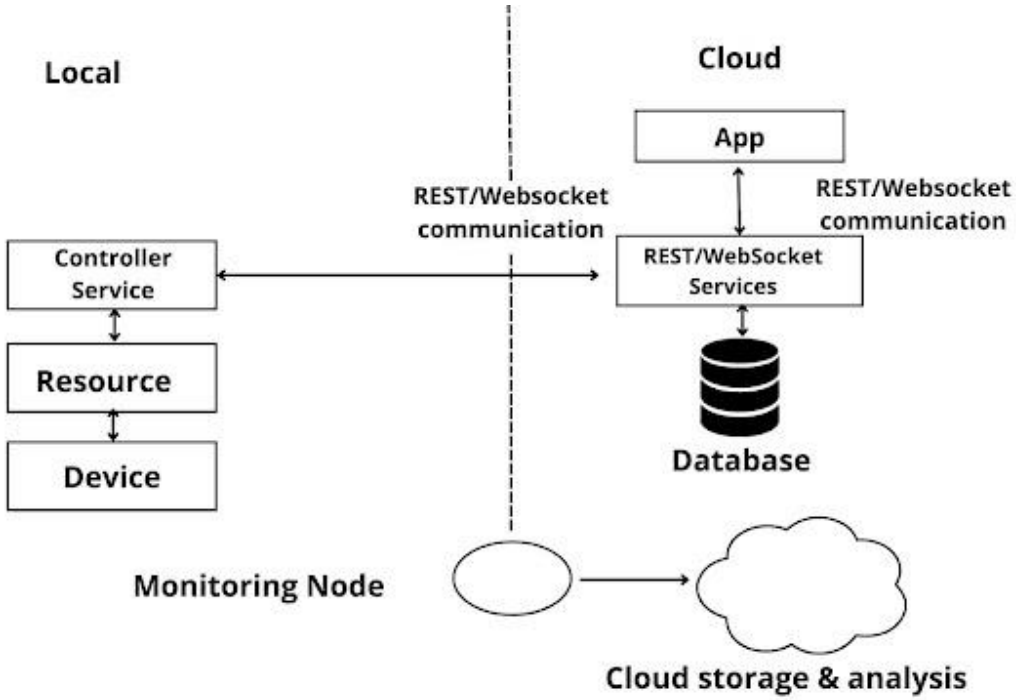


IoT Level-2

IoT level 3

At this level, the application is cloud-based. A single node monitors the environment and stores data in the cloud. This is suitable where data is comprehensive and analysis 1 computationally intensive.

Example: A node is monitoring a package using devices like an accelerometer and gyroscope. These devices track vibration levels. controller service sends sensor data to the cloud in the rear time using WebSocket APL. Data is stored in the cloud and visualized using a cloud-based application. The analysis component triggers an alert if vibration levels cross a threshold.

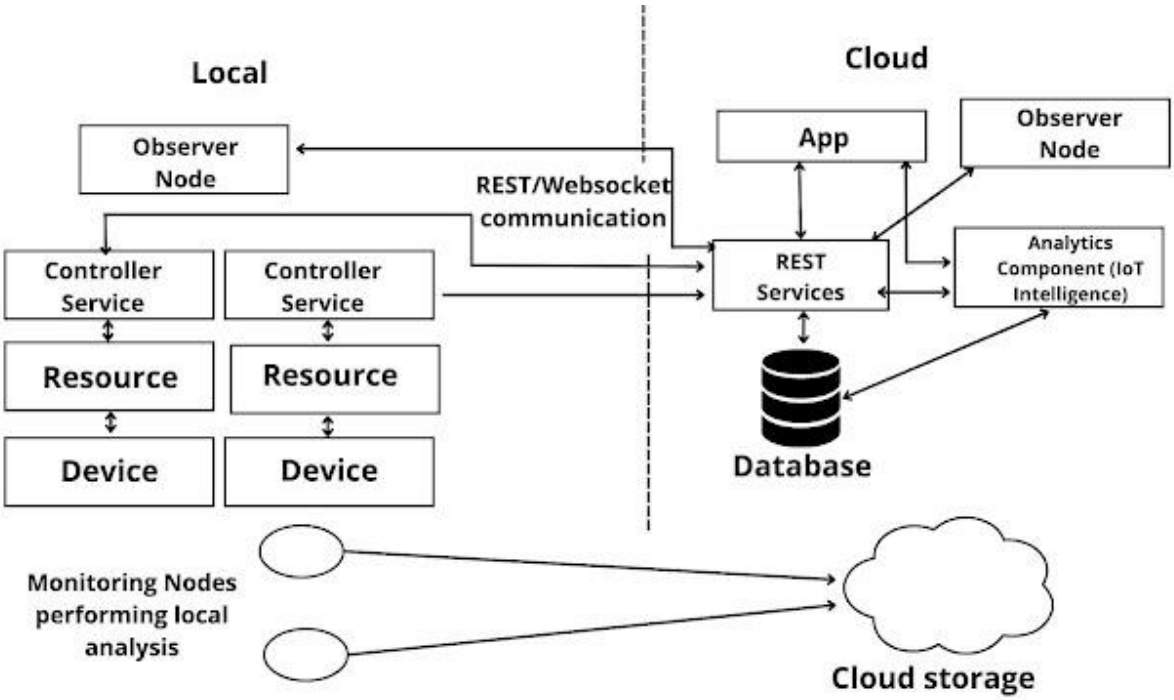


IoT Level-3

IoT level 4

At this level, Multiple nodes collect information and store it in the cloud. Local and rent server nodes are used to grant and receive information collected in the cloud from various devices. Observer nodes can process information and use it for applications but not perform control functions, This level is the best solution where data involvement is big, requirement analysis is comprehensive and multiple nodes are required,

Example: Analysis is done on the cloud and the entire IoT system has monitored the cloud using an application. Noise monitoring of an area requires various nodes to function independently of each other. Each has its own controller service. Data is stored in a cloud database.

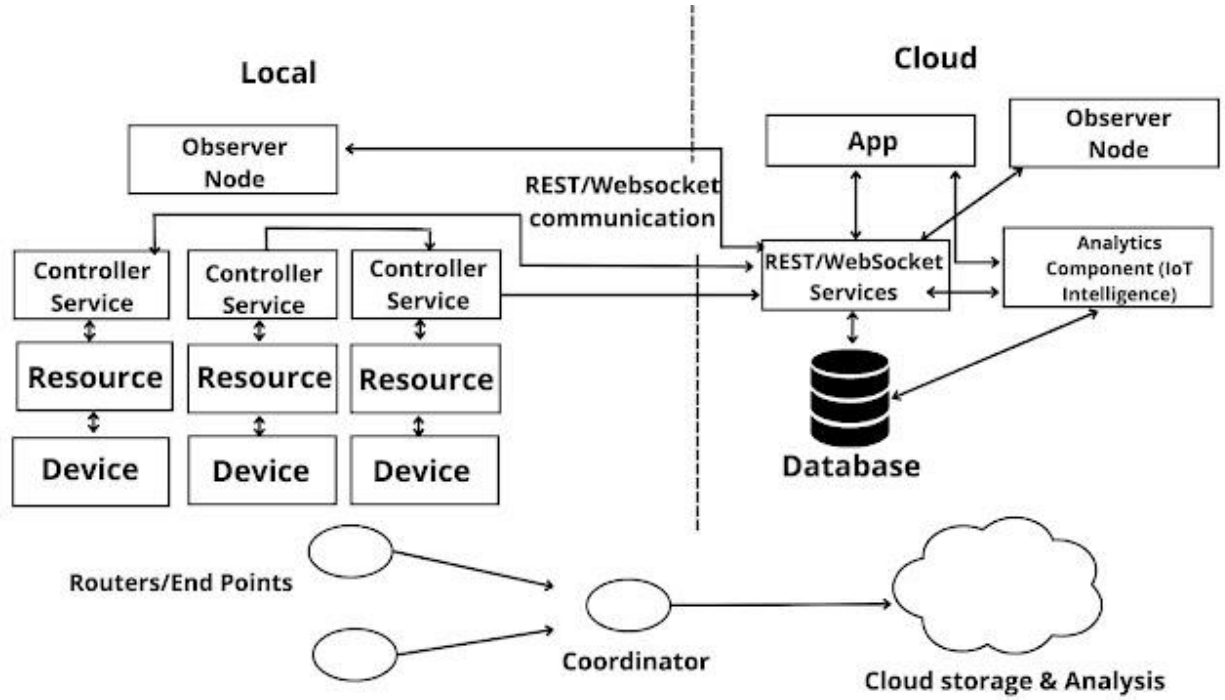


IoT Level-4

IoT level 5

In this level Nodes present locally are of two types end odes and coordinator nodes End nodes collect data and perform sensing or actuation or both. Coordinator nodes collect data from end nodes and send it to the cloud. Data is stored and analyzed in the cloud. This level is best for WSN, where the data involved is big and the requirement analysis is comprehensive.

Example: A monitoring system has various components: end nodes collect various data from the environment and send it to the coordinator node. The coordinator node acts as a gateway and allows the data to be transferred to cloud storage using REST API. The controller service on the coordinator node sends data to the cloud.



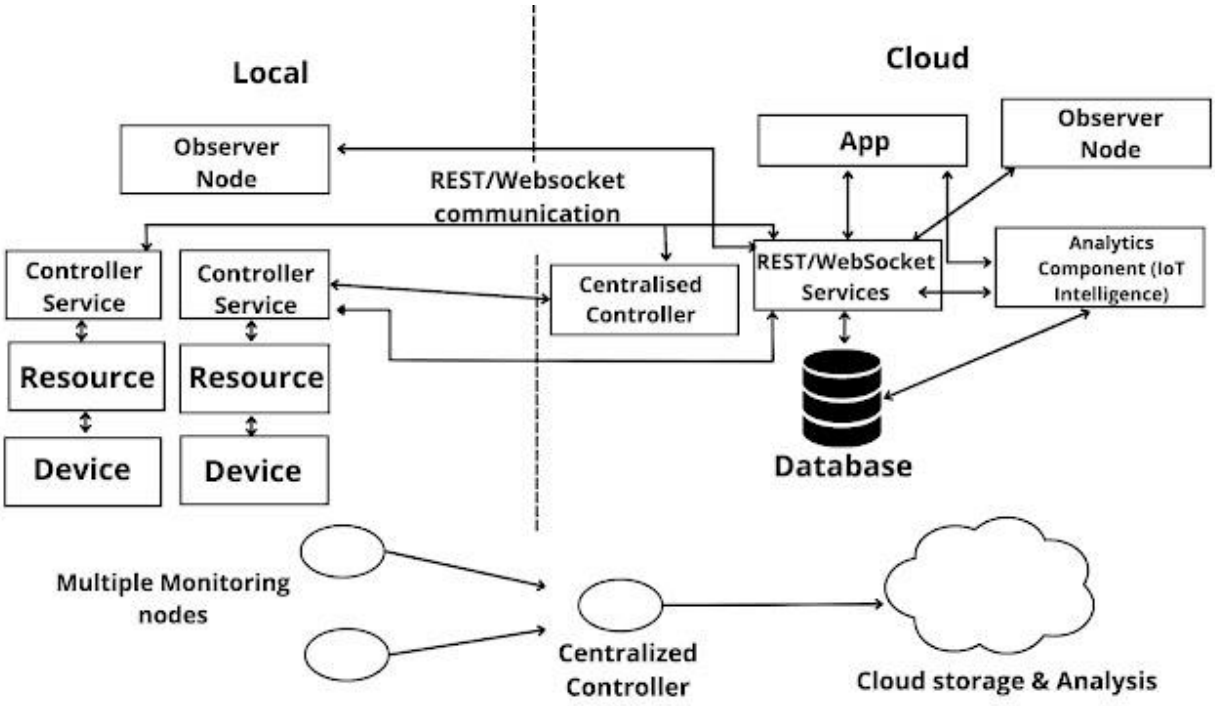
IoT Level-5

IoT Level 6

At this level, the application is also cloud-based and data is stored in the cloud-like of levels. Multiple independent end nodes perform sensing and actuation and send d to the cloud. The analytics components analyze the data and store the results in the cloud database. The results are visualized with a cloud-based application. The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.

Example: Weather monitoring consists of sensors that monitor different aspects of the system. The end nodes send data to cloud storage. Analysis of components, applications, and storage areas in the cloud. The centralized controller controls all nodes and provides inputs.

IoT Level-6



Domain-specific IoTs (Internet of Things) refer to IoT applications and devices that are designed and tailored to meet the specific requirements and challenges of a particular industry or field. Unlike generic IoT devices that can be applied across various domains, domain-specific IoT solutions are optimized for a particular use case, offering specialized features, capabilities, and protocols.

Here are some examples of domain-specific IoT applications in various industries:

1.Industrial IoT (IIoT): In manufacturing and industrial settings, IoT devices are used to monitor and control machinery, optimize production processes, and improve overall efficiency. Sensors and actuators are deployed to gather data on equipment performance, energy usage, and environmental conditions.

2. Smart Agriculture: IoT is employed in agriculture to monitor soil conditions, crop health, and weather patterns. Farmers can use this data to make informed decisions about irrigation, fertilization, and pest control, leading to increased crop yields and resource efficiency.

3. **Healthcare IoT: In the healthcare sector, IoT devices are used for remote patient monitoring, asset tracking, and inventory management. Wearable devices and sensors can collect real-time health data, providing healthcare professionals with valuable insights for personalized patient care.

4. Smart Cities: IoT technologies are integrated into urban infrastructure to enhance city services and improve the quality of life for residents. Examples include smart streetlights, waste management systems, and traffic monitoring to optimize city planning and resource allocation.

5. **Retail IoT:In the retail industry, IoT devices are used for inventory management, supply chain optimization, and enhancing the overall customer experience. RFID tags, sensors, and beacons help retailers track product availability, manage stock levels, and provide personalized promotions to customers.

6. **Energy Management:** IoT is applied in energy systems to monitor and optimize energy consumption. Smart grids, connected appliances, and energy-efficient devices contribute to better energy management, reducing waste and lowering overall energy costs.

7. **Connected Vehicles:** Automotive IoT involves the integration of sensors and connectivity features in vehicles to enable functions such as real-time navigation, predictive maintenance, and vehicle-to-vehicle communication for enhanced safety.

8. **Smart Buildings:** IoT devices are used in building management systems to monitor and control lighting, HVAC systems, security, and occupancy. This helps in creating energy-efficient and comfortable environments while reducing operational costs.

9. Environmental Monitoring: IoT devices play a crucial role in monitoring and collecting data on environmental conditions such as air and water quality, helping to address issues related to pollution and climate change.

10.Logistics and Supply Chain: IoT is used to track and manage the movement of goods throughout the supply chain. This includes real-time tracking of shipments, monitoring of storage conditions, and optimizing transportation routes for efficiency.

IoTivity is an open-source project that provides a framework for building IoT (Internet of Things) solutions. It is designed to enable seamless device-to-device connectivity and communication in IoT ecosystems. The IoTivity stack architecture is structured to support interoperability between devices, regardless of the underlying hardware or communication protocols. Here are the key components and layers of the IoTivity stack architecture:

1. Device Layer:

-Device Abstraction: This layer abstracts the underlying hardware details and provides a standardized representation of devices. It includes device models and capabilities to ensure interoperability.

- Device Discovery: IoTivity supports device discovery mechanisms, allowing devices to find and identify each other on the network.

2. Connectivity Abstraction Layer:

-Connectivity Abstraction: This layer abstracts communication protocols and networking details, providing a uniform interface for device communication. It enables devices to communicate over different network protocols, such as Wi-Fi, Ethernet, or Bluetooth.

3. Resource-Encapsulation Layer:

- **Resource Model:** IoTivity utilizes a resource-oriented model, where each device exposes its functionalities as resources. Resources can be sensors, actuators, or other features of a device.

- **Resource Discovery:** Devices can discover and interact with resources on other devices, promoting a standardized way of accessing and controlling device functionalities.

4. Common Services Layer:

- **Security:** IoTivity includes security mechanisms to ensure secure communication between devices. This may involve authentication, encryption, and secure key exchange.

- **Data Management:** This layer handles data management tasks, including serialization and deserialization of data for communication between devices.

- **Error Handling:** Common services provide error handling mechanisms to manage and report errors that may occur during device interactions.

5. Middleware Layer:

- **Message Handling:** Middleware is responsible for handling messages between devices. It manages the communication flow, including message routing and delivery.

- **Event Handling**: Middleware supports event-driven programming, allowing devices to respond to changes in the environment or the state of other devices.

6. Application Framework Layer:

- **APIs (Application Programming Interfaces):** This layer provides APIs for application developers to build IoT applications. It abstracts the lower layers, simplifying the development of applications that interact with IoT devices.

- **Application Logic:** IoTivity supports the development of applications that can control and monitor devices. Application logic can include rules, automation, and user interfaces.

7. Application Layer:

- **User Interfaces:** This layer involves the development of user interfaces for end-users to interact with IoT devices. This can include mobile apps, web interfaces, or other types of user interfaces.

- **Application-Specific Logic**: Applications at this layer implement specific functionalities and use cases, such as home automation, industrial monitoring, or healthcare applications.

What is Hadoop?

Hadoop is an open source software programming framework for storing a large amount of data and performing the computation. Its framework is based on Java programming with some native code in C and shell scripts.

Hadoop is an open-source software framework that is used for storing and processing large amounts of data in a distributed computing environment. It is designed to handle big data and is based on the MapReduce programming model, which allows for the parallel processing of large datasets.

Hadoop has two main components:

HDFS (Hadoop Distributed File System): This is the storage component of Hadoop, which allows for the storage of large amounts of data across multiple machines. It is designed to work with commodity hardware, which makes it cost-effective.

YARN (Yet Another Resource Negotiator): This is the resource management component of Hadoop, which manages the allocation of resources (such as CPU and memory) for processing the data stored in HDFS.

Hadoop also includes several additional modules that provide additional functionality, such as Hive (a SQL-like query language), Pig (a high-level platform for creating MapReduce programs), and HBase (a non-relational, distributed database).

Hadoop is commonly used in big data scenarios such as data warehousing, business intelligence, and machine learning. It's also used for data processing, data analysis, and data mining. It enables the distributed processing of large data sets across clusters of computers using a simple programming model.

History of Hadoop

Apache Software Foundation is the developers of Hadoop, and it's co-founders are **Doug Cutting** and **Mike Cafarella**. It's co-founder Doug Cutting named it on his son's toy elephant. In October 2003 the first paper release was Google File System. In January 2006, MapReduce development started on the Apache Nutch which consisted of around 6000 lines coding for it and around 5000 lines coding for HDFS. In April 2006 Hadoop 0.1.0 was released.

Hadoop is an open-source software framework for storing and processing big data. It was created by Apache Software Foundation in 2006, based on a white paper written by Google in 2003 that described the Google File System (GFS) and the MapReduce programming model. The Hadoop framework allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. It is used by many organizations, including Yahoo, Facebook, and IBM, for a variety of purposes such as data warehousing, log processing, and research. Hadoop has been widely adopted in the industry and has become a key technology for big data processing.

Features of hadoop:

1. it is fault tolerance.

2. it is highly available.

3. it’s programming is easy.

4. it have huge flexible storage.

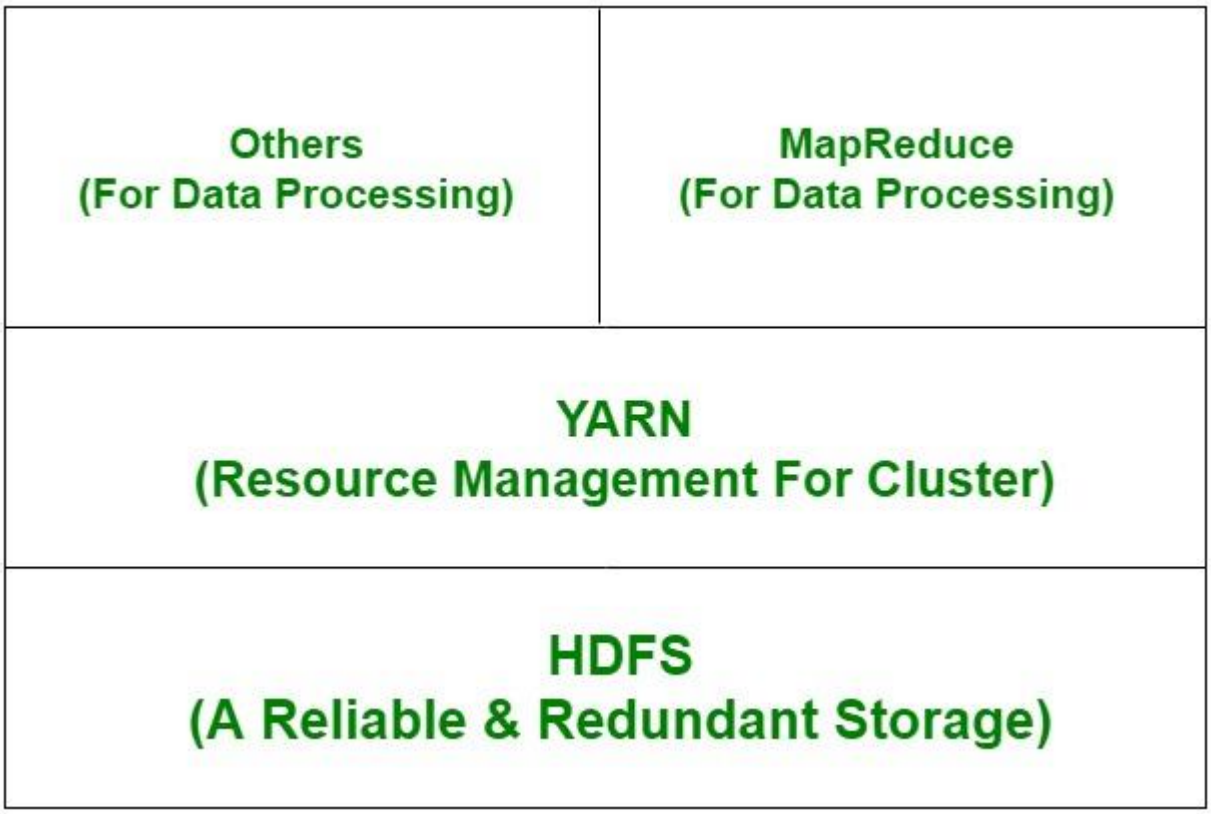
5. it is low cost.

Hadoop has several key features that make it well-suited for big data processing:

Distributed Storage: Hadoop stores large data sets across multiple machines, allowing for the storage and processing of extremely large amounts of data.
Scalability: Hadoop can scale from a single server to thousands of machines, making it easy to add more capacity as needed.
Fault-Tolerance: Hadoop is designed to be highly fault-tolerant, meaning it can continue to operate even in the presence of hardware failures.
Data locality: Hadoop provides data locality feature, where the data is stored on the same node where it will be processed, this feature helps to reduce the network traffic and improve the performance
High Availability: Hadoop provides High Availability feature, which helps to make sure that the data is always available and is not lost.
Flexible Data Processing: Hadoop’s MapReduce programming model allows for the processing of data in a distributed fashion, making it easy to implement a wide variety of data processing tasks.
Data Integrity: Hadoop provides built-in checksum feature, which helps to ensure that the data stored is consistent and correct.
Data Replication: Hadoop provides data replication feature, which helps to replicate the data across the cluster for fault tolerance.
Data Compression: Hadoop provides built-in data compression feature, which helps to reduce the storage space and improve the performance.
YARN: A resource management platform that allows multiple data processing engines like real-time streaming, batch processing, and interactive SQL, to run and process data stored in HDFS.

Hadoop Distributed File System

It has distributed file system known as HDFS and this HDFS splits files into blocks and sends them across various nodes in form of large clusters. Also in case of a node failure, the system operates and data transfer takes place between the nodes which are facilitated by HDFS.



HDFS

Advantages of HDFS: It is inexpensive, immutable in nature, stores data reliably, ability to tolerate faults, scalable, block structured, can process a large amount of data simultaneously and many more. **Disadvantages of HDFS:** It's the biggest disadvantage is that it is not fit for small quantities of data. Also, it has issues related to potential stability, restrictive and rough in nature. Hadoop also supports a wide range of software packages such as Apache Flumes, Apache Oozie, Apache HBase, Apache Sqoop, Apache Spark, Apache Storm, Apache Pig, Apache Hive, Apache Phoenix, Cloudera Impala.

Some common frameworks of Hadoop

. Hive- It uses HiveQl for data structuring and for writing complicated MapReduce in HDFS.

. Drill- It consists of user-defined functions and is used for data exploration.
. Storm- It allows real-time processing and streaming of data.
. Spark- It contains a Machine Learning Library(MLlib) for providing enhanced machine learning and is widely used for data processing. It also supports Java, Python, and Scala.
. Pig- It has Pig Latin, a SQL-Like language and performs data transformation of unstructured data.
. Tez- It reduces the complexities of Hive and Pig and helps in the running of their codes faster.

Hadoop framework is made up of the following modules:

. Hadoop MapReduce- a MapReduce programming model for handling and processing large data.
. Hadoop Distributed File System- distributed files in clusters among nodes.
. Hadoop YARN- a platform which manages computing resources.
. Hadoop Common- it contains packages and libraries which are used for other modules.

Advantages and Disadvantages of Hadoop

Advantages:

Ability to store a large amount of data.

High flexibility.

Cost effective.

High computational power.

Tasks are independent.

Linear scaling.

Hadoop has several advantages that make it a popular choice for big data processing:

Scalability: Hadoop can easily scale to handle large amounts of data by adding more nodes to the cluster.

Cost-effective: Hadoop is designed to work with commodity hardware, which makes it a cost-effective option for storing and processing large amounts of data.

Fault-tolerance: Hadoop's distributed architecture provides built-in fault-tolerance, which means that if one node in the cluster goes down, the data can still be processed by the other nodes.

Flexibility: Hadoop can process structured, semi-structured, and unstructured data, which makes it a versatile option for a wide range of big data scenarios.

Open-source: Hadoop is open-source software, which means that it is free to use and modify. This also allows developers to access the source code and make improvements or add new features.

Large community: Hadoop has a large and active community of developers and users who contribute to the development of the software, provide support, and share best practices.

Integration: Hadoop is designed to work with other big data technologies such as Spark, Storm, and Flink, which allows for integration with a wide range of data processing and analysis tools.

Disadvantages:

Not very effective for small data.

Hard cluster management.

Has stability issues.

Security concerns.

Complexity: Hadoop can be complex to set up and maintain, especially for organizations without a dedicated team of experts.

Latency: Hadoop is not well-suited for low-latency workloads and may not be the best choice for real-time data processing.

Limited Support for Real-time Processing: Hadoop's batch-oriented nature makes it less suited for real-time streaming or interactive data processing use cases.

Limited Support for Structured Data: Hadoop is designed to work with unstructured and semi-structured data, it is not well-suited for structured data processing

Data Security: Hadoop does not provide built-in security features such as data encryption or user authentication, which can make it difficult to secure sensitive data.

Limited Support for Ad-hoc Queries: Hadoop's MapReduce programming model is not well-suited for ad-hoc queries, making it difficult to perform exploratory data analysis.

Limited Support for Graph and Machine Learning: Hadoop's core component HDFS and MapReduce are not well-suited for graph and machine learning workloads, specialized components like Apache Graph and Mahout are available but have some limitations.

Cost: Hadoop can be expensive to set up and maintain, especially for organizations with large amounts of data.

Data Loss: In the event of a hardware failure, the data stored in a single node may be lost permanently.

Data Governance: Data Governance is a critical aspect of data management, Hadoop does not provide a built-in feature to manage data lineage, data quality, data cataloging, data lineage, and data audit.

What is Apache?

Apache is free and open-source software of web server that is used by approx **40% of websites** all over the world. Apache HTTP Server is its official name. It is developed and maintained by the **Apache Software Foundation**. Apache permits the owners of the websites for serving content over the web. It is the reason why it is known as a "**web server**." One of the most reliable and old versions of the Apache web server was published in 1995.

If someone wishes to visit any website, they fill-out the name of the domain in their browser address bar. The web server will bring the requested files by performing as the virtual delivery person.

Web Server Meaning

Mail servers, database servers, web servers, and **file servers** use different types of server software. All these applications may access a lot of files saved on the physical server and apply them for many objectives.

The aim of the web servers is to deliver websites over the internet. It behaves as a middleman among the client machines and servers to achieve that aim. It can pull the content through the server over every user request. Also, it delivers this request to the web.

Backward Skip 10sPlay VideoForward Skip 10s

One of the most critical tasks of any web server is to provide services to various different users of the web at the same time. Web servers execute files specified in a different type of programming languages like **Java**, **Python**, **PHP**, and many others. Web servers turn these files into static HTML files. It provides services to these files within the web server browser. Web browser can be defined as a tool liable for decent client-server communication.

Working of Apache

Apache is not any physical server; it is software that executes on the server. However, we define it as a web server. Its objective is to build a connection among the website visitor browsers (Safari, Google Chrome, Firefox, etc.) and the server. Apache can be defined as cross-platform software, so it can work on Windows servers and UNIX.

When any visitor wishes for loading a page on our website, the homepage, for instance, or our "About Us" page, the visitor's browser will send a request on our server. Apache will return a response along with each requested file (images, files, etc.). The client and server communicate by HTTP protocol, and Apache is liable for secure and smooth communication among t both the machines.

Apache is software that is highly **customizable**. It contains the module-based structure. Various modules permit server administrators for turning additional functionality off and on. Apache includes modules for caching, security, password authentication, URL rewriting, and other purposes. Also, we can set up our own configuration of the server with the help of a file known as **.htaccess**. It is a supported configuration file of Apache.