Unit- 4
Data Analytics and Supporting Services

Dr. Madhukrishna Priyadarsini
Department of CSE

Outline:

- Structured VS Unstructured data
- Data in motion VS Data in rest
- Role of machine learning
- NoSQL databases
- Hadoop Eco system(Apache kafka, Apache spark)
- Edge streaming Analytics
- Network Analytics
- Xively cloud for IoT
- Python web application framework(Django, AWS, NETCONF-YANG)

Structured VS Unstructured data

- **Structured data**: follows a model that defines how the data is presented or organized; Either in RDBMS or simple spread sheet.

Ex: banking transactions, router configuration

- Structured data is easily formatted, stored, queried, and processed; that's why used for making business decisions.
- A wide array of data analytics tools are readily available for processing this type of data. Ex: Microsoft Excel, Tableau

- **Unstructured data:** lacks a logical schema for understanding and decoding the data through traditional programming means. Ex: speech, text, image, video
- Any data that does not fit neatly into a predefined data model is unstructured data.
- 80% of the business data is unstructured, so data analytics methods such as cognitive computing, machine learning are applied to the business data.
- Smart objects in IoT generate both structured and unstructured data.

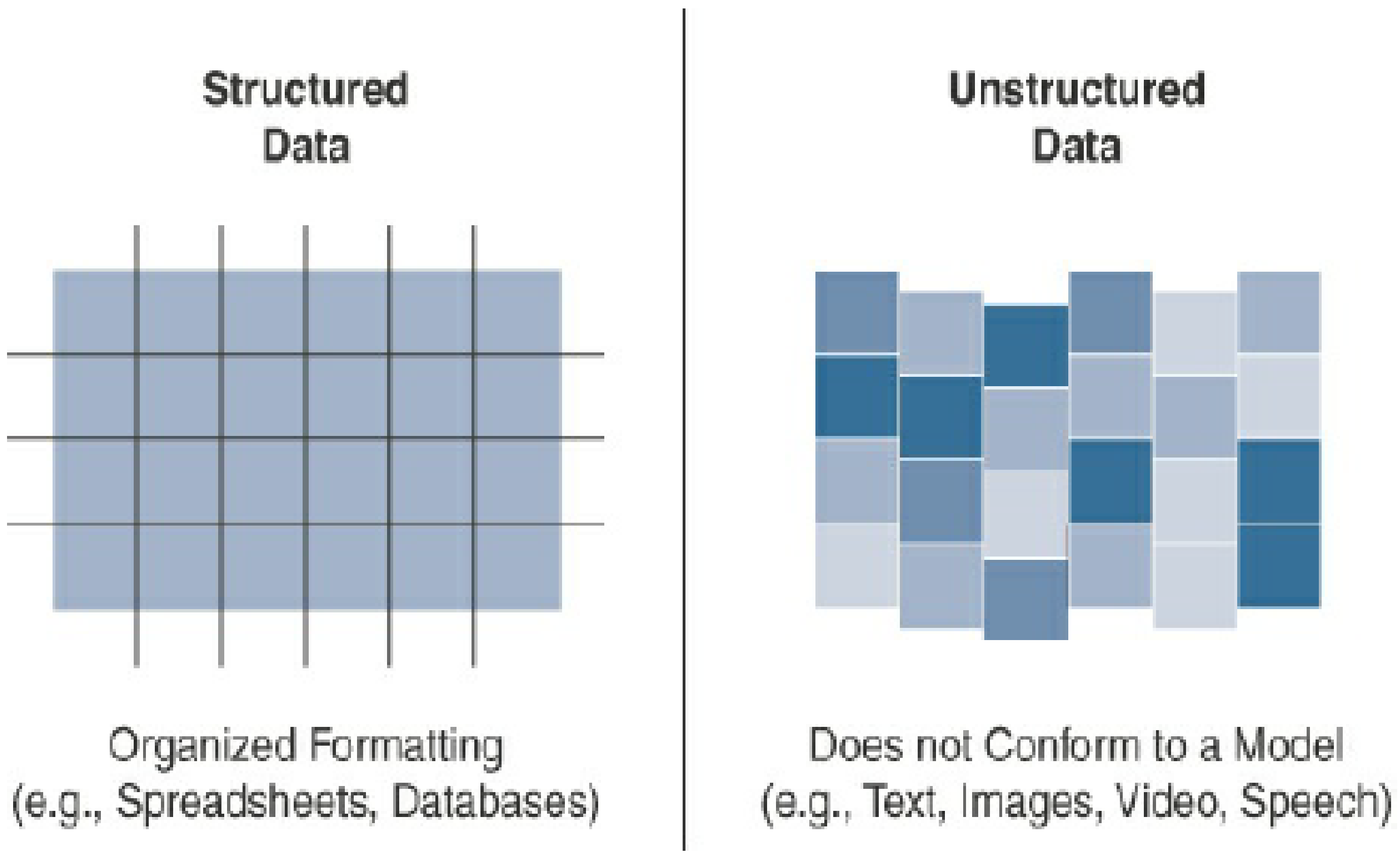

Structured Data
Unstructured Data
Organized Formatting
(e.g., Spreadsheets, Databases)
Does not Conform to a Model
(e.g., Text, Images, Video, Speech)

Figure 7-2 *Comparison Between Structured and Unstructured Data*

Data in Motion VS Data at Rest

- Data in IoT network is either in transit (“data in motion”) or being held or stored (“data at rest”).

Ex: data in motion: client/server communication(web browsing, email, file transfers)

Data at rest: data saved to hard drive, storage array, USB drive

- From an IoT perspective data from smart objects are considered as “data in motion”, and when it is processed, filtered, forwarded at fog node or data center, is treated as “data at rest” .
- Tools used for processing “data in motion” are Spark, Storm, Flink
- “Data at rest” are normally processed in IoT brokers or storage array at the data center. Ex: Hadoop; it not only helps with data processing, but also data storage.

IoT Data Analytics Overview

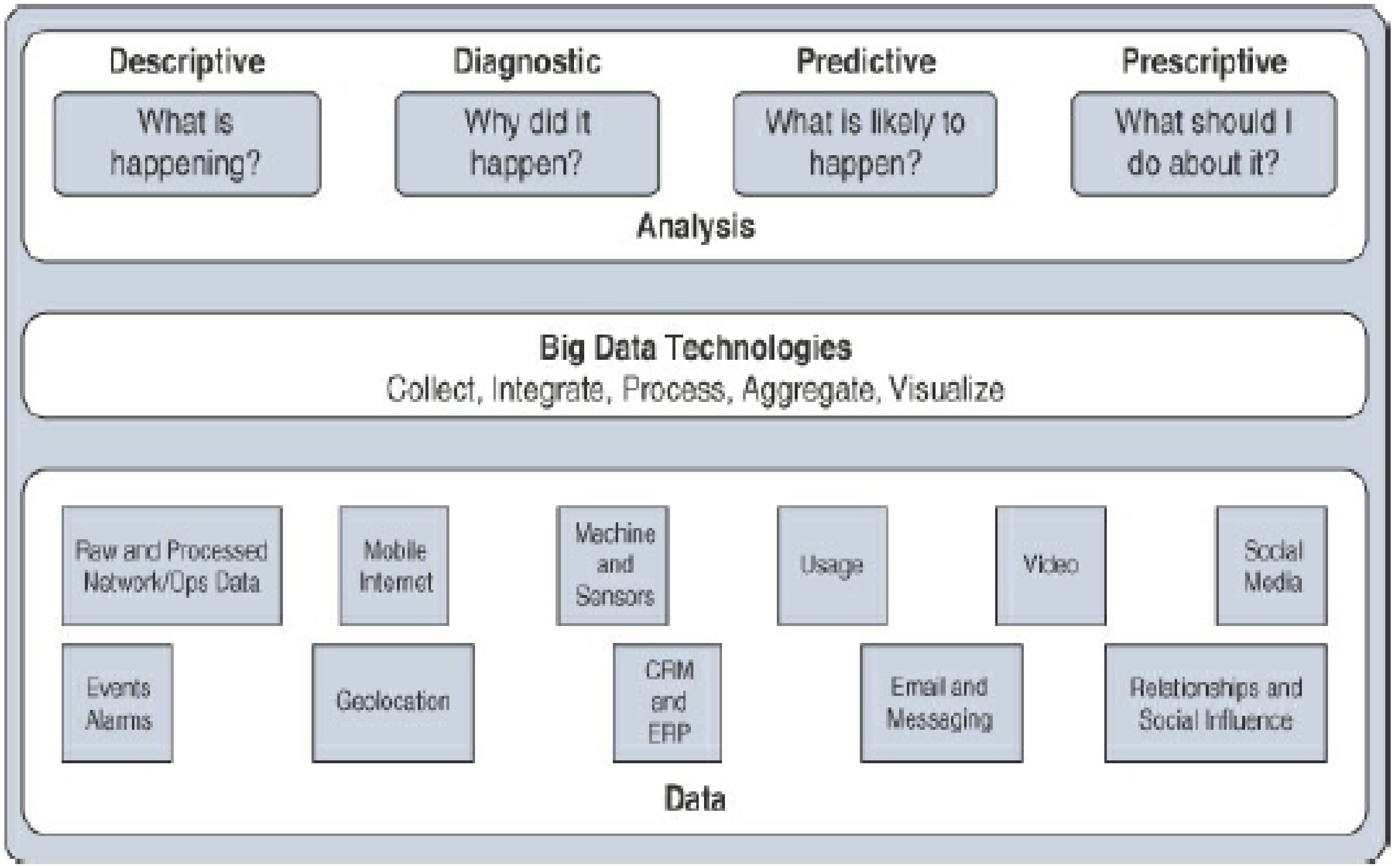


Figure 7-3 *Types of Data Analysis Results*

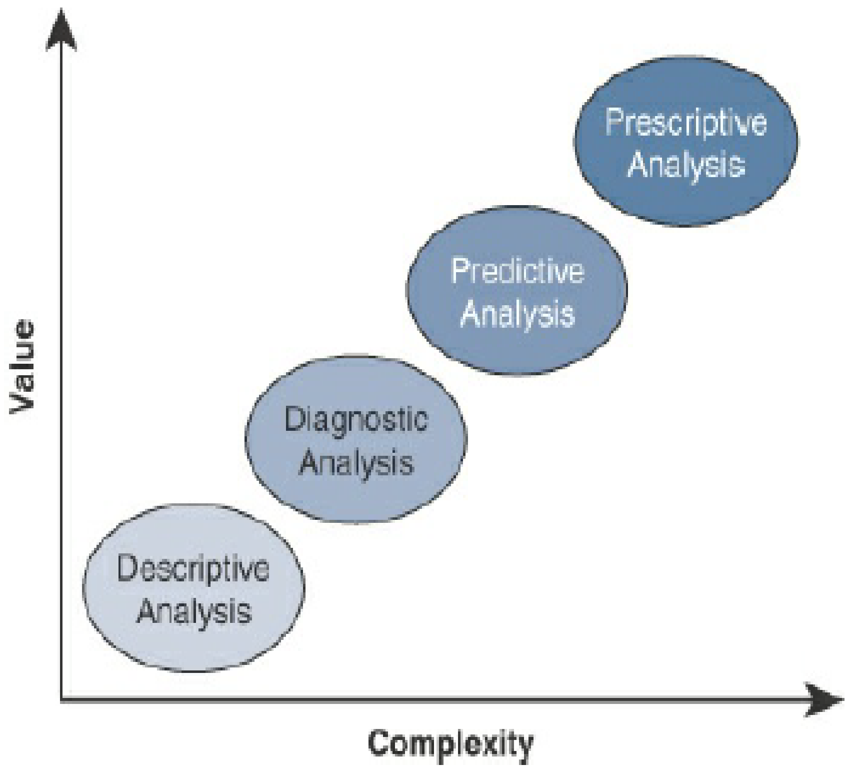


Figure 7-4 *Application of Value and Complexity Factors to the Types of Data Analysis*

Data Analytics Challenges:

1. Scailing Problem-> large number of smart objects continually send data and the database grows very quickly which requires more hardware and architecture changes in order to solve the performance issues.
2. Volatility of data-> in relational database changing schema can slow or stop the database from operating. Due to lack of flexibility, revisions to the schema must be kept at a minimum. In IoT, the data model/schema is likely to change and evolve over time. A dynamic schema is required to solve the issue.

Role of Machine Learning

- The main goal of IoT generated data is to make sense, which leads to specialized tools and algorithms to find the data relationship. This brings the concept of “Machine Learning” in IoT.
- Machine Learning Overview: Any computer that uses rules to make decisions belong to the paradigm of “Artificial Intelligence”, and machine learning is part of the paradigm.

Ex: An app that can find your parked car, GPS calculating speed of vehicle

- Machine learning is concerned with any process where the computer needs to receive a set of data that is processed to help perform a task with more efficiency.
- ML is divided in to two categories: supervised and unsupervised learning

Supervised Learning

- The machine is trained with input for which there is a known correct answer.
- Training-> classification
- Testing-> validation
- Learning process is not about classifying in two or more categories, but about finding the correct value.
- Regression predicts numeric values; whereas classification predicts categories

Unsupervised Learning

- In some cases supervised learning is not the best method for a machine to help with a human decision.
- The method used to cluster unlabeled/untagged data is unsupervised learning. There is no good or bad answer known in advance.
- It is the variation from a group behaviour that allows the computer to learn something different.

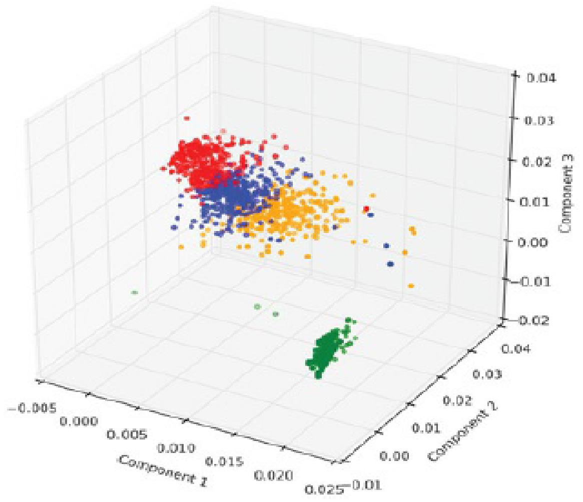
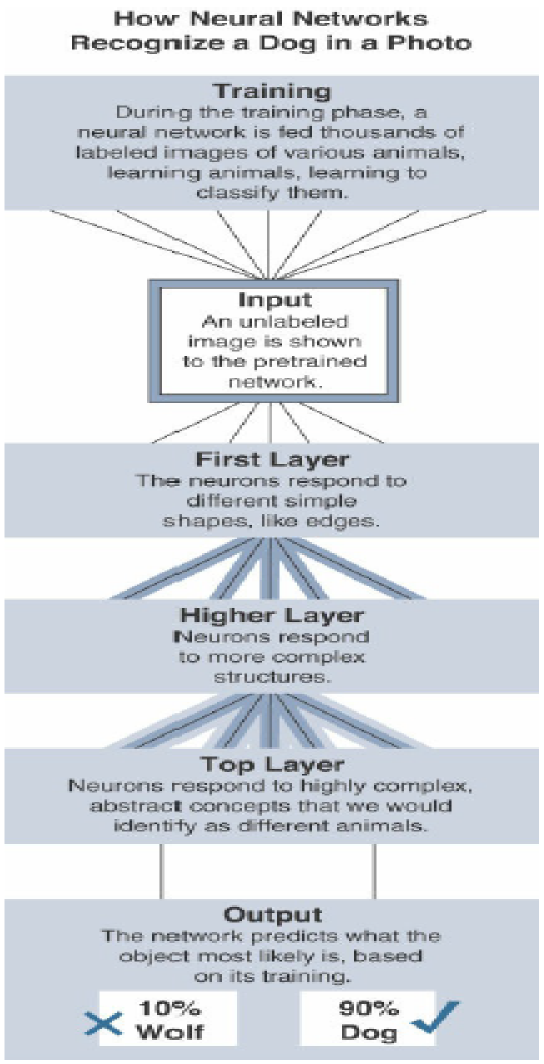


Figure 7-5 *Clustering and Deviation Detection Example*

Neural Networks

- Neural networks are ML methods that mimic the way human brain works.
- The information regarding an object goes through different algorithms/units, where each one is charge of processing an aspect of the information.
- The resulting value of one unit computation can be used directly/ fed into another unit for further processing to occur.
- There are different layers of processing in a neural network. The efficiency is that each unit processes a simple test and therefore computation is quiet fast.

Neural network processing example


How Neural Networks Recognize a Dog in a Photo
Training
During the training phase, a neural network is fed thousands of labeled images of various animals, learning animals, learning to classify them.
Input
An unlabeled image is shown to the pretrained network.
First Layer
The neurons respond to different simple shapes, like edges.
Higher Layer
Neurons respond to more complex structures.
Top Layer
Neurons respond to highly complex, abstract concepts that we would identify as different animals.
Output
The network predicts what the object most likely is, based on its training.
10% Wolf
90% Dog

Machine learning operations are divided into two subgroups:

1. Local learning->data is collected and processed locally; either in sensors or gateways
2. Remote learning->data is collected and sent to a central processing unit, where it is processed

- Inherited learning->optimizes local operations

Common applications of ML and IoT revolve around 4 major domains:

a. Monitoring-> smart objects monitor the environment where they operate. ML can be used to monitor early failure conditions
b. Behaviour control->Monitoring commonly works in conjunction with behaviour control which aims at taking corrective actions based on thresholds.
c. Operations optimization->Analyzing data can lead to overall process improvement
d. Self-healing, self-optimizing->The system dynamically combines and monitors new parameters and automatically implements new optimization whenever required

Databases for IoT context

- Massively parallel processing databases-> built on the concept of relational data warehouses, but designed to be much faster, reduced query processing time. Both data and processing are distributed across multiple systems. It operates in a "shared nothing" fashion, with each node containing local processing, memory, and storage. It allows common SQL tools and applications to work with the database.

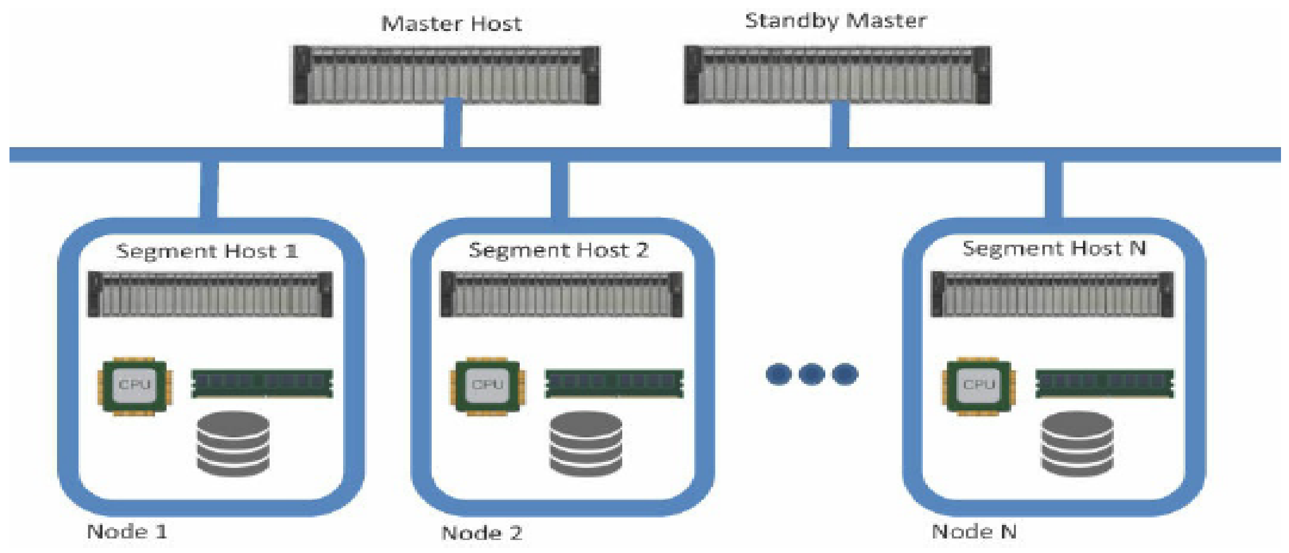


Figure 7-7 *MPP Shared-Nothing Architecture*

NoSQL(“not only SQL”) Databases

- It supports semi-structured and unstructured data, in addition to the structured data handled by warehouses and MPPs.
- It is an umbrella term that encompasses several different types of databases, including the following:

1. Document stores->stores semi-structured data such as XML, JSON. Document stores mainly have query engines and indexing features that allow for many optimized queries.
2. Key-value stores->stores associative arrays where a key is paired with an associated value. These databases are easy to build and easy to scale.
3. Wide-column stores->stores similar to key-value store, but the formatting of the values can vary from row to row, even in the same table.
4. Graph stores->organized based on the relationships between elements. It is used for social media, NLP where the connection between data are very relevant.

- It was developed to support high-velocity, urgent data requirements of modern web applications that do not require repeated use.
- Its intent is to ingest rapidly changing server logs and clickstream data(user data while browsing a website) generated by web-scale applications that did not fit into rows and columns.
- It is built to scale horizontally, allowing the database to span multiple hosts, and can even be distributed geographically.
- NoSQL can expand to other distributed data systems, where additional nodes are managed by a master node. This provision makes NoSQL to hold and handle sensor data associated with smart objects.
- Key-value stores and document-stores are the best fit databases for IoT data.
- Key-value stores are capable of handling indexing and persistence simultaneously at a high rate. It is best for used in time-series data sets.
- Database schema is changed quickly for key-value stores, so more flexible. It also allows to query the database through an API, making it easy to integrate with other data management applications.

Hadoop

- It is the most popular choice as a data repository and processing engine.
- The original intent for Hadoop was to index millions of websites and quickly return search results for opensource search engines. Initially it had two key components:

1. Hadoop Distributed File System (HDFS)-> A system for storing data across multiple nodes.
2. MapReduce-> A distributed processing engine that splits a large task into smaller ones that can be run in parallel

Both of them are providing foundations for other projects in the current version of Hadoop.

- Hadoop leverages local processing, memory, and storage to distribute tasks and provide a scalable storage system for data.
- Both MapReduce and HDFS store and process massive amount of data and thus leverages resources from all nodes in the cluster.

For HDFS, the resource leverage is handled by specialized nodes in the cluster, namely; NameNodes and DataNodes

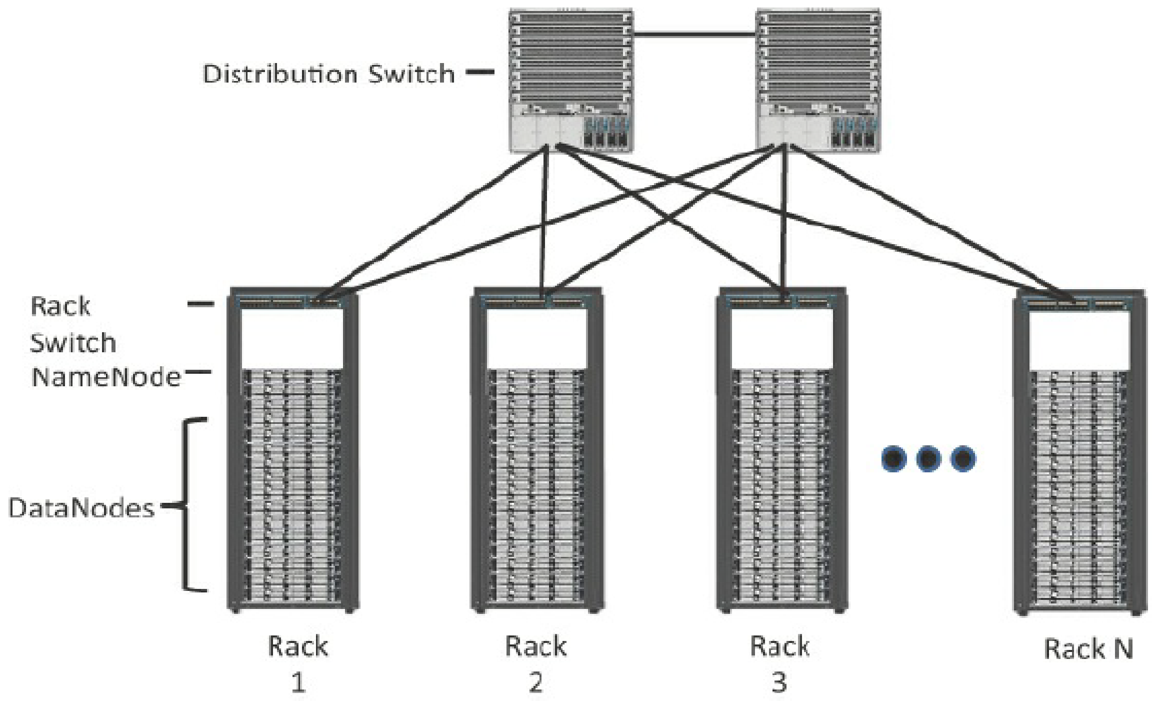


Figure 7-8 *Distributed Hadoop Cluster*

- NameNodes-> these nodes co-ordinate where the data is stored, and replicated using a map. All interactions with HDFS is done using a primary NameNode (active), and using a secondary NameNode (standby) the changes during a failure event of primary is notified.

- The NameNodes take write requests from clients and distributes those files across the available nodes in configurable block sizes, usually 64MB or 128 MB blocks. It is also responsible to instruct the DataNodes where the replication should occur.
- DataNodes-> these are the servers where the data is stored at the direction of NameNode. There are multiple DataNodes present in a Hadoop cluster to store data. Data blocks are distributed across several nodes and often are replicated 3,4, or more times across nodes for redundancy.
- Once the data is written to one of the DataNodes, that DataNode selects two additional nodes based on the replication policies, to ensure data redundancy across the cluster. RAID technique is not used here, because both NameNodes and DataNodes coordinate block-level redundancy.

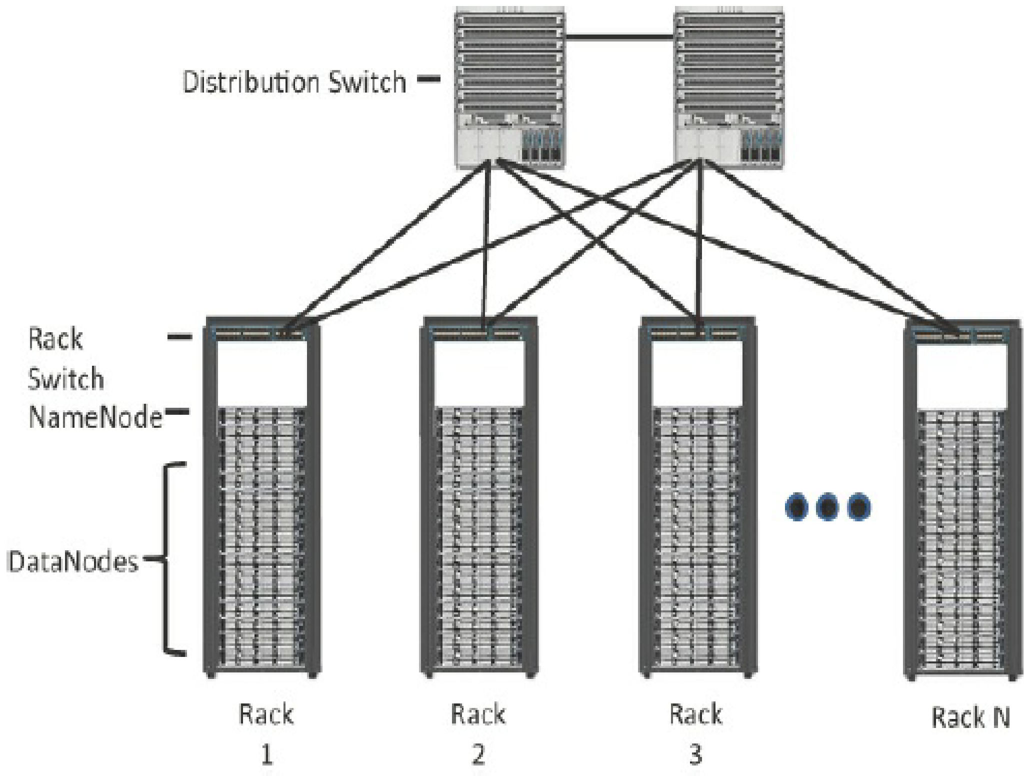


Figure 7-8 *Distributed Hadoop Cluster*

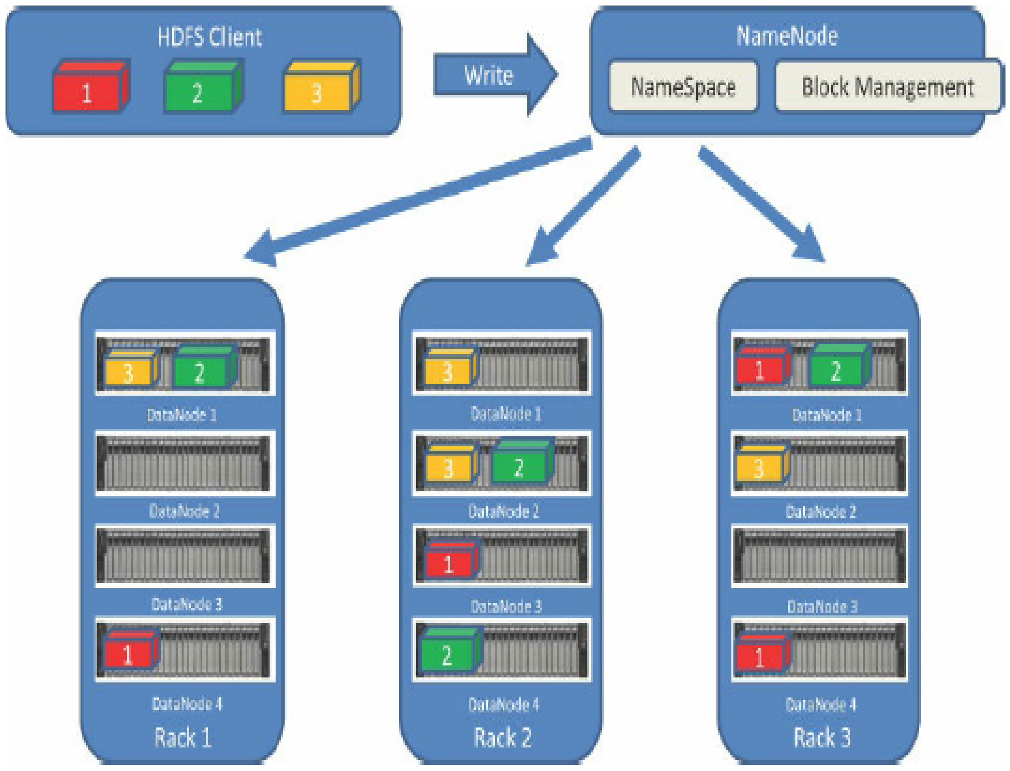


Figure 7-9 *Writing a File to HDFS*

MapReduce leverages to batch process the data stored on the cluster nodes.

- Batch processing is the process of running a scheduled query across the historical data stored in HDFS. A query is broken down into smaller tasks and distributed across all the nodes running MapReduce in a cluster.
- It is mostly used for understanding patterns and trending in historical sensor or machine data.
- One drawback of MapReduce is time; depending on the query's complexity, the result takes seconds or minutes to return.
- For real-time processing MapReduce is not the right data processing engine.

The Hadoop Ecosystem

- From the initial release (2011) to till date, many projects have been developed to add incremental functionality to Hadoop and collectively known as the ***Hadoop Ecosystem.***
- Hadoop now comprises more than 100 software projects under the Hadoop umbrella, capable of every element in the data life cycle, from collection, to storage, to processing, to analysis and visualization.
- Each of these individual projects is a unique piece of the overall data management solution; some of them are:

Apache Kafka:

Messaging systems are designed to accept data, or messages, from where the data is generated and deliver the data to stream processing engines such as Spark Streaming or Storm. Apache Kafka is a distributed publisher-subscriber messaging system that is built to be scalable and fast. It is composed of topics, or message brokers, where producers write data and consumers read data from these topics.

- Due to the distributed nature of *Kafka,* it can run in a clustered configuration that can handle many producers and consumers simultaneously and exchange of information between nodes, allowing topics to be distributed over multiple nodes.
- The goal of *Kafka* is to provide a simple way to connect to data sources and allow consumers to connect to that data in the way they would like

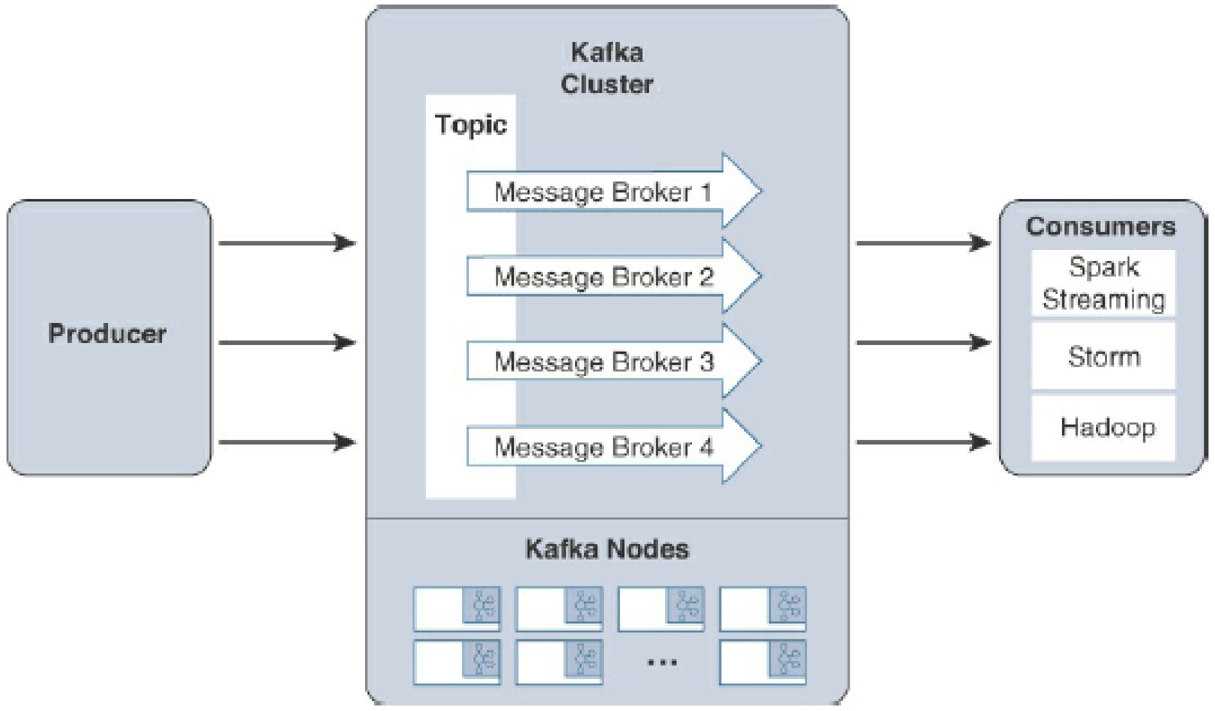


Figure 7-10 *Apache Kafka Data Flow*

Apache Spark

- It is an "in-memory" data analytics platform designed to accelerate processes in the Hadoop ecosystem.
- In-memory->The data read and write operation is moved to high speed memory, which introduces low latency, speeds the batch processing job, and allows the real-time processing of events.
- Real-time processing is done through a component of Apache Spark project called "**Spark Streaming".** Spark streaming is responsible for taking live streamed data from a messaging system, and dividing it into smaller microbatches, called <u>Discretized streams/Dstreams.</u>
- The Spark processing engine is able to operate on these smaller pieces of data, allowing rapid insights into the data, and subsequent actions.
- Spark is providing benefits to systems that control safety and security of personnel, time-sensitive processes in the manufacturing space, and infrastructure control in traffic management.

- In the Hadoop ecosystem, different projects are similar and have significant overlap with other projects. For example; Apache Spark is used for distributed streaming analytics and batch processing both. Similarly Apache Storm and Apache Flink are two other Hadoop ecosystem projects designed for distributed streaming and used in many IoT use cases.
- Storm can pull data from Kafka and process it in a real-time fashion, and so can Flink.

Lambda Architecture:

- One architecture is currently used by many IoT use cases to process and storage data using multiple technologies, querying data (data in motion and rest) using multiple Hadoop ecosystem projects is called *Lambda Architecture.*
- Lambda architecture is having two layers for ingesting data(batch and stream) and one layer for providing combined data(serving). These layers allow to operate on data independently, focusing on the key attributes for which they are designed and optimized.
- Data is taken from a message broker, commonly Kafka, processed in each layer in parallel, and the resulting data is delivered into a data store where additional processing is done.

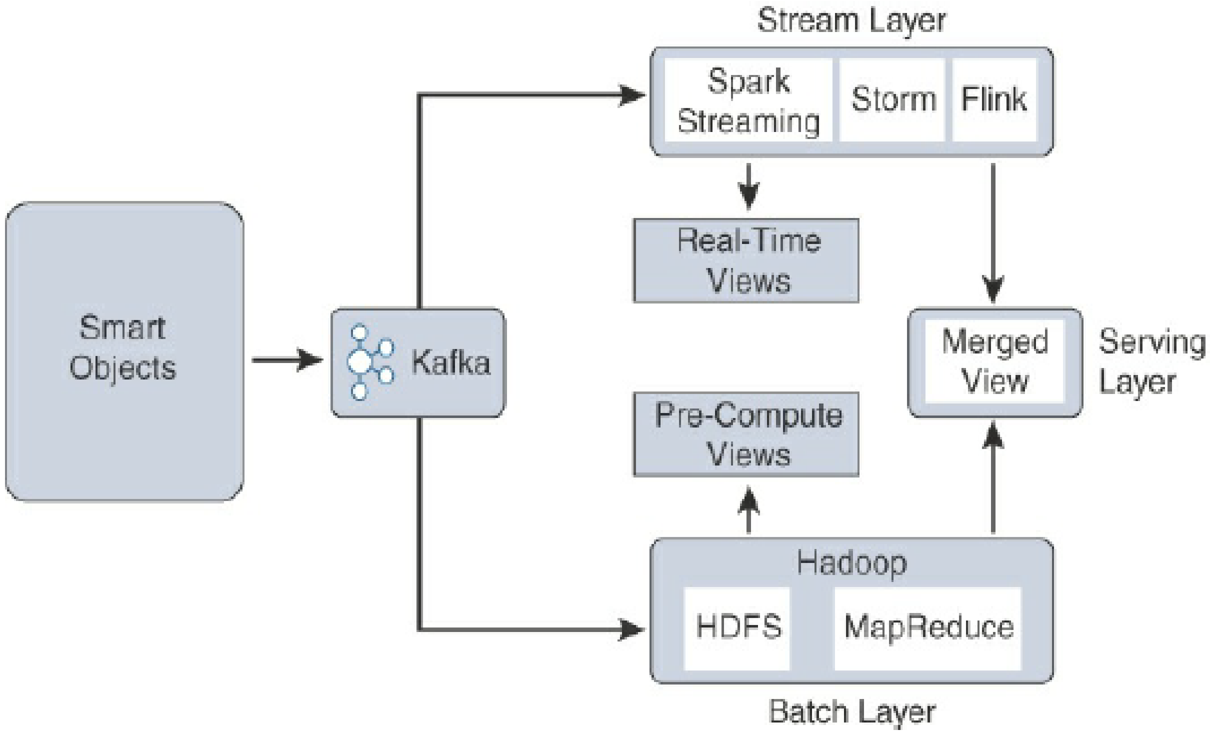


Figure 7-11 *Lambda Architecture*

- Stream Layer-> responsible for real-time processing of events. Spark streaming, Storm, Flink technologies are used to ingest, process, and analyze data on this layer. Alerting and automated actions can be triggered on events that require rapid response.

- Batch Layer-> It consists of batch processing engine and data store. Ex: MapReduce, HDFS, MPPs, NoSQL, data warehouses etc.
- Serving Layer-> It is a data store or mediator that decides which of the ingest layers to query based on the expected result or view into the data. If an aggregate or historic view is requested, it may invoke the batch layer. If real-time analytics are needed it may invoke the stream layer. Serving layer is often used by the data consumers to access both layers simultaneously.

- Lambda architecture can provide a robust system for collecting and processing massive amounts of data, and flexibility to analyze that data at different rates.
- Due to the processing and storage requirements, the Lambda architecture is deployed either in data centers or in clouds, which reduces its effectiveness for the processing systems present milliseconds away from the sensors generating data. For this distributed Edge-processing architectures are used.

Edge Streaming Analytics

- The vast quantities of data are generated on the fly and often need to be analyzed and responded immediately. The huge data are generated on the edge sensors and transferred to the cloud using the network bandwidth. Here the streaming of data analytics techniques come into picture.

Ex: Formula One racing

- The key values of edge streaming analytics include the followings:

1. Reducing data at the edge
2. Analysis and response at the edge
3. Time sensitivity

Edge analytics core functions:

Edge analytics continually processes streaming flows of data in motion.

Streaming analytics at the edge can be broken down into 3 simple stages:

- **Raw input data->** raw data from sensors into the analytics processing units
- **Analytics processing units (APU)->** it filters and combines data streams, organizes them by time windows, and performs various analytical functions
- **Output streams->** the output data is organized into insightful streams and is used to influence the behavior of the smart objects, passed for storage and further processing in the cloud. Communication with the cloud happens through a standard publisher/subscriber messaging protocol, such as MQTT

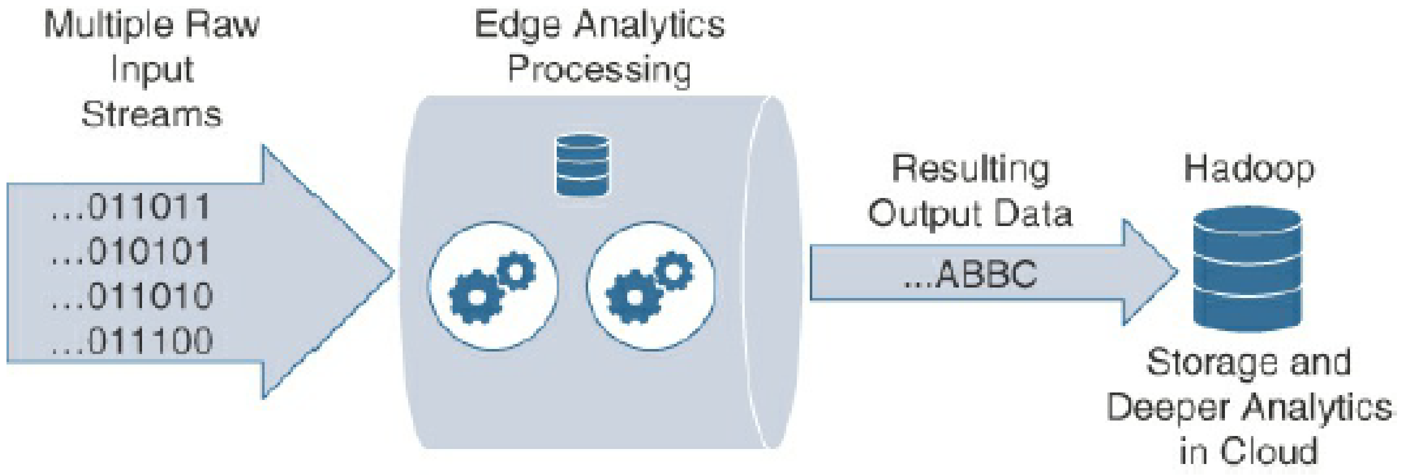
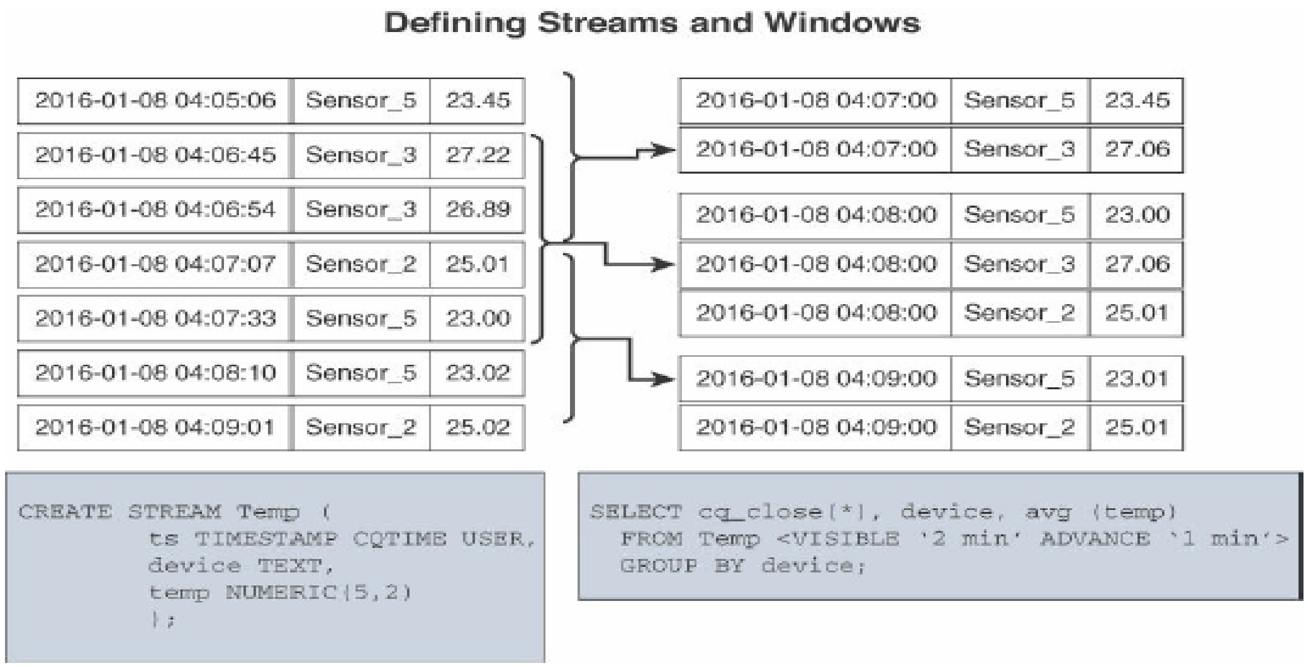

Multiple Raw Input Streams
...011011
...010101
...011010
...011100
Edge Analytics Processing
Resulting Output Data
...ABBC
Hadoop
Storage and Deeper Analytics in Cloud

Figure 7-12 *Edge Analytics Processing Unit*

In order to perform analysis in real-time, the APU needs to perform the following functions:

1. **Filter:** the filter function identifies the information that is considered relevant
2. **Transform:** In data warehousing, Extract, Transform, and Load(ETL) operations are used to manipulate the data structure into a form that can be used for other purpose. Analogous to data warehouse ETL operations, in streaming analytics, once the data is filtered it needs to be formatted for processing.
3. **Time:** As the data is streaming in real-time, timing constraints plays a vital role.



Defining Streams and Windows

2016-01-08 04:05:06	Sensor_5	23.45
2016-01-08 04:06:45	Sensor_3	27.22
2016-01-08 04:06:54	Sensor_3	26.89
2016-01-08 04:07:07	Sensor_2	25.01
2016-01-08 04:07:33	Sensor_5	23.00
2016-01-08 04:08:10	Sensor_5	23.02
2016-01-08 04:09:01	Sensor_2	25.02

2016-01-08 04:07:00	Sensor_5	23.45
2016-01-08 04:07:00	Sensor_3	27.06
2016-01-08 04:08:00	Sensor_5	23.00
2016-01-08 04:08:00	Sensor_3	27.06
2016-01-08 04:08:00	Sensor_2	25.01
2016-01-08 04:09:00	Sensor_5	23.01
2016-01-08 04:09:00	Sensor_2	25.01

```
CREATE STREAM Temp (
        ts TIMESTAMP CQTIME USER,
        device TEXT,
        temp NUMERIC(5,2)
        );
```

```
SELECT cq_close(*), device, avg (temp)
  FROM Temp <VISIBLE '2 min' ADVANCE '1 min'>
  GROUP BY device;
```

Figure 7-13 *Example: Establishing a Time Window for Analytics of Average Temperature from Sensors*

4. Correlate: Streaming data analytics becomes most useful when multiple data streams are combined from different types of sensors. Correlate goes beyond combining real-time data streams. It is also relating real-time measurements with pre-existing or historical data.

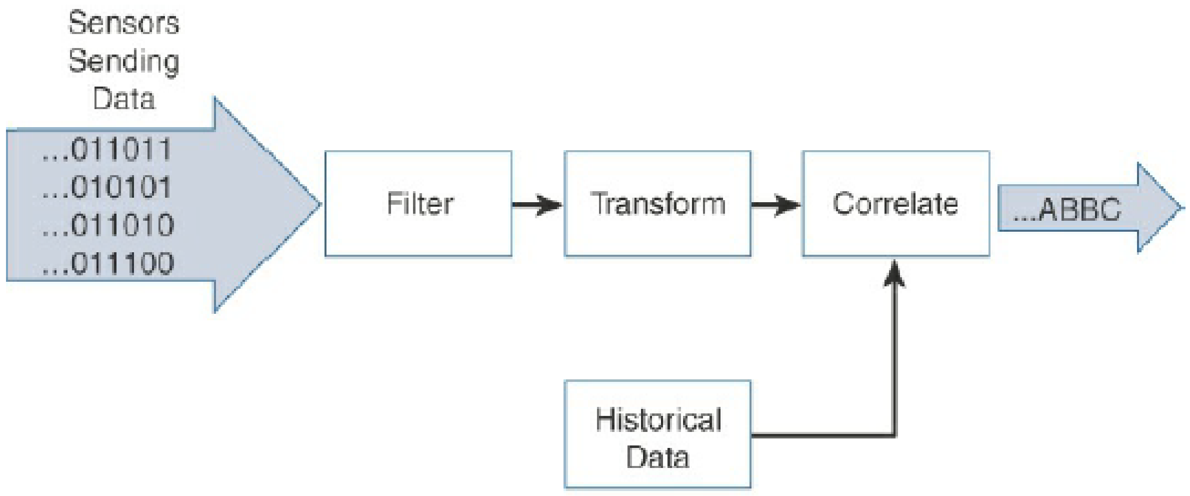


Figure 7-14 *Correlating Data Streams with Historical Data*

5. Match Patterns: Once the data streams are properly cleaned, transformed, and correlated with other live streams as well as historical data sets, pattern matching operations are used to gain deeper insights to the data. Patterns can be simple or complex defined by the criteria in the application. Machine learning may be leveraged to identify these patterns.

6. Improve business intelligence: Ultimately, the value of edge analytics is in the improvements to the business intelligence that were not previously available. Over time, the resulting changes in business logic can produce improvements in basic operations.

Distributed Analytics Systems:

Depending on the application and network architecture, analytics can happen at any point throughout the IoT system. Streaming analytics may be performed at the edge, fog, or in the cloud data centre. There is no hard-and-fast rules dictating where analytics can be done, but there are few guiding principles. Sometimes it is not efficient to stand at the edge devices to know the importance of the data, rather stepping back and finding out a wider view to get a overall importance.

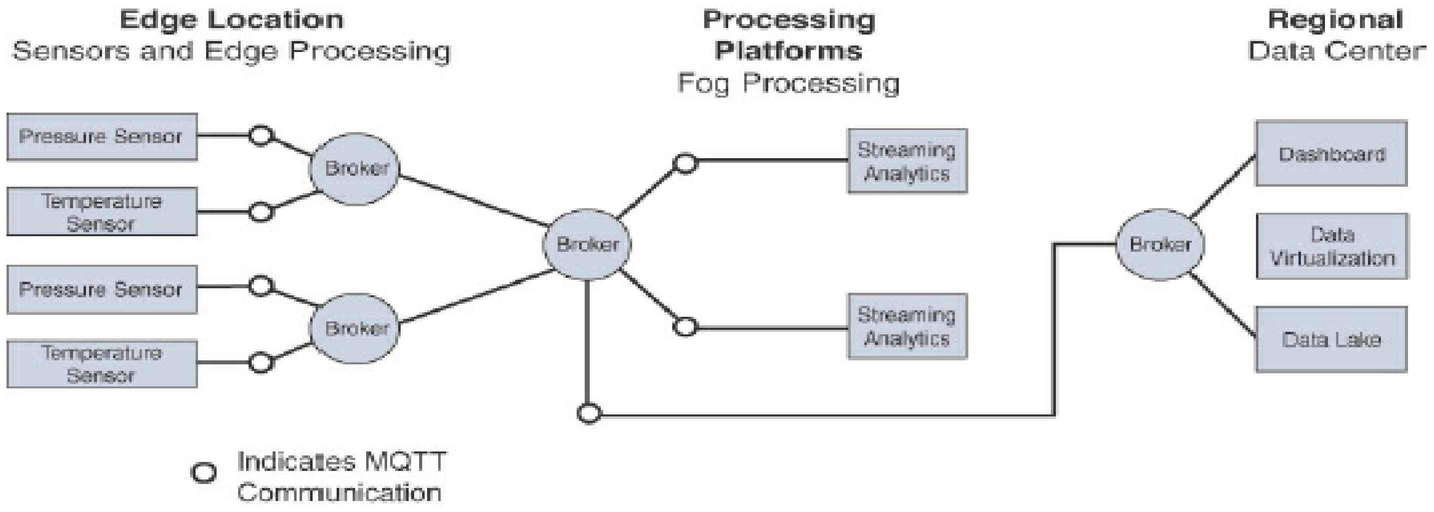


Figure 7-15 *Distributed Analytics Throughout the IoT System*

Network Analytics

- Unlike data analytics systems are concerned with finding patterns in the data generated by endpoints, network analytics is concerned with discovering patterns in the communication flows from a network traffic perspective.
- It analyzes details of communications patterns made by protocols and correlate this across the network. It differentiates normal and abnormal (excessive congestion, intrusive malware etc) behaviour of the network.

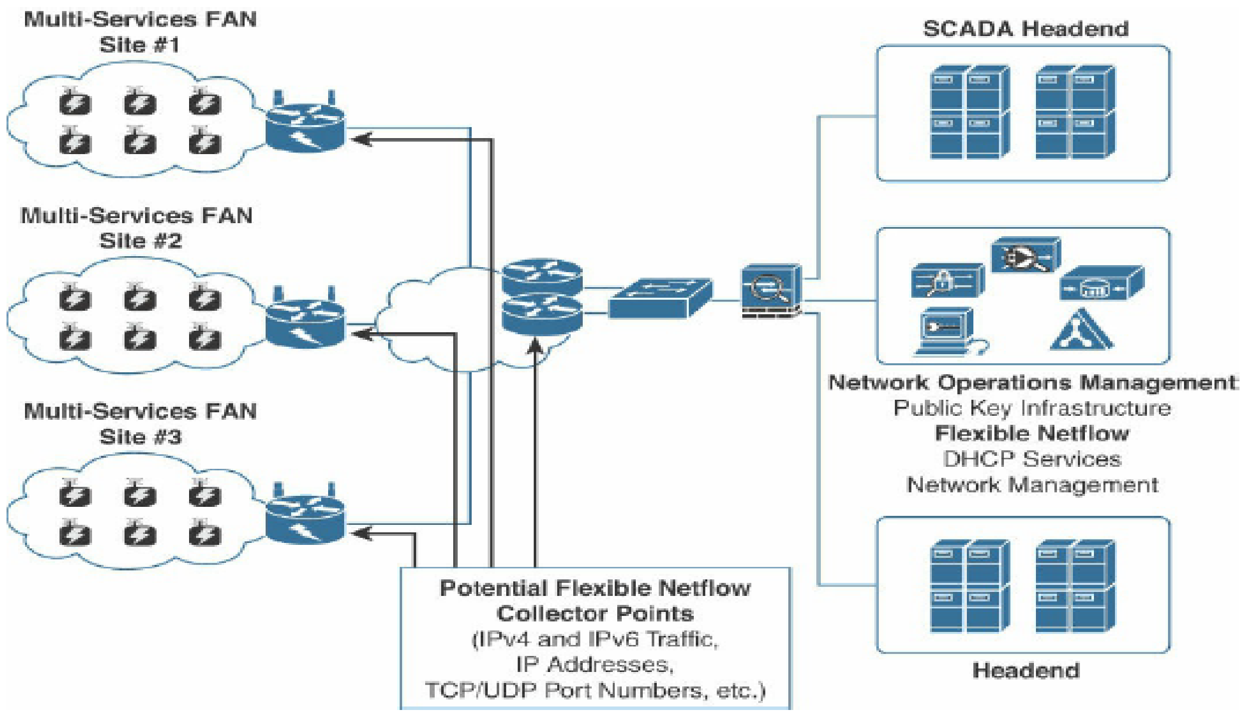

Multi-Services FAN Site #1
Multi-Services FAN Site #2
Multi-Services FAN Site #3
SCADA Headend
Network Operations Management
Public Key Infrastructure
Flexible Netflow
DHCP Services
Network Management
Headend
Potential Flexible Netflow Collector Points
(IPv4 and IPv6 Traffic,
IP Addresses,
TCP/UDP Port Numbers, etc.)

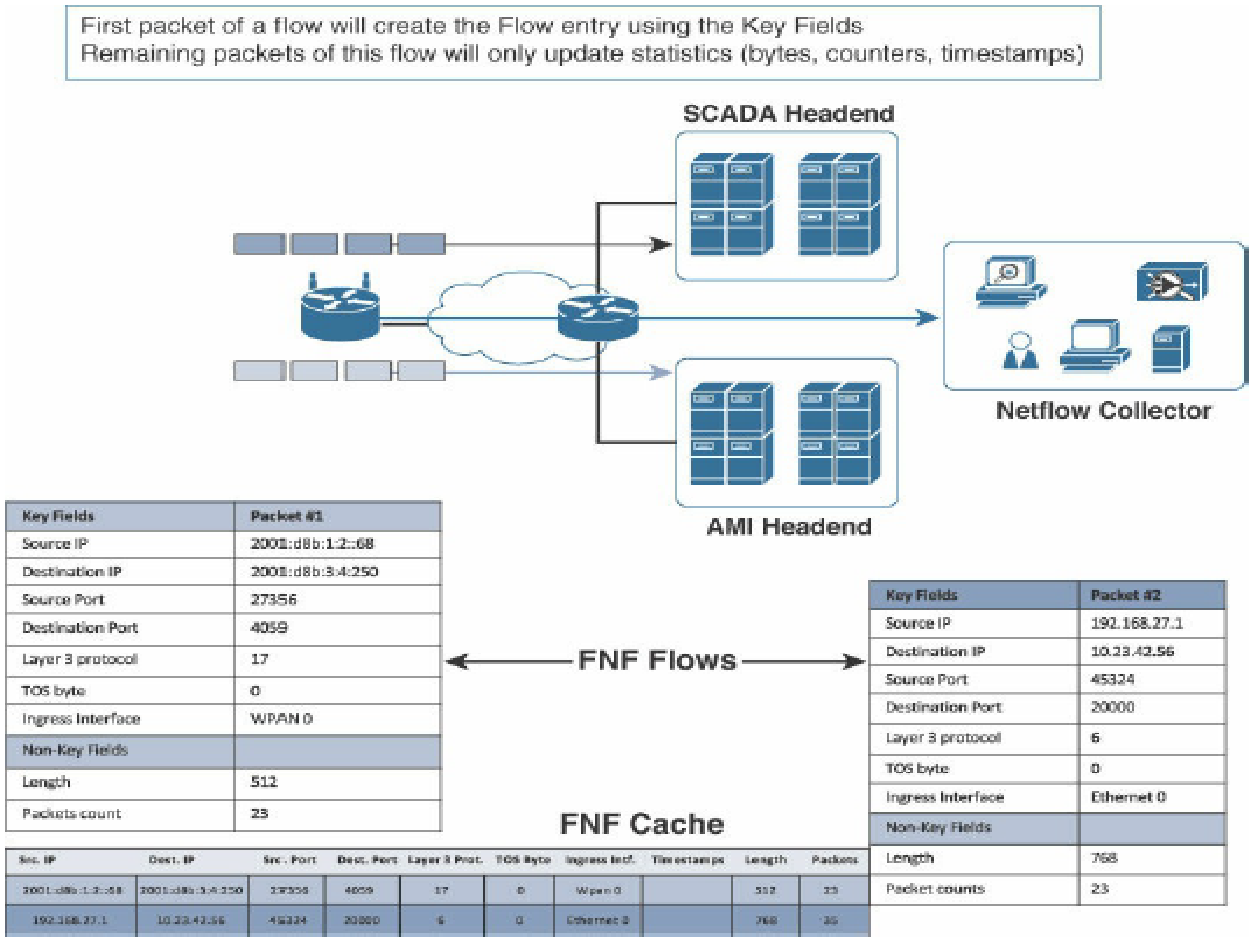
Figure 7-16 *Smart Grid FAN Analytics with NetFlow Example*

- Network analytics offer capabilities to cope with capacity planning for scalable IoT deployment as well as security monitoring in order to detect abnormal traffic volume and patterns for both centralized and decentralized architectures.
- To monitor network flows, IoT standards and protocols allow pervasive characterization of IP traffic flows, identification of source and destination address, data timing and volume, and application types within a network infrastructure. Flow statistics can be collected at different locations in the network.
- After collection of data in the known format, it can be sent to external network analytics tools that delivers unique services such as security and performance monitoring, capacity planning.
- Network analytics supports different network management services such as:

1. Network traffic monitoring and profiling->IPV4 and IPV6 networkwide traffic volume and pattern analysis
2. Application traffic monitoring and profiling->application layer protocols including MQTT, CoAP, DNP3
3. Capacity planning-> can track and anticipate IoT traffic growth, and help in planning of upgrades
4. Security Analysis->any change in the network behaviour may indicate a cyber security event
5. Accounting->flow monitoring can be used to monitoring and optimizing the bills
6. Data warehousing and data mining->flow data can be warehoused for later retrieval and analysis

Flexible NetFlow Architecture (FNF)

- FNF is a flow technology developed by Cisco Systems that is widely deployed all over the world.


First packet of a flow will create the Flow entry using the Key Fields
Remaining packets of this flow will only update statistics (bytes, counters, timestamps)
SCADA Headend
Netflow Collector
AMI Headend
FNF Flows
FNF Cache

Key Fields	Packet #1
Source IP	2001:d8b:1:2::68
Destination IP	2001:d8b:3:4:250
Source Port	27356
Destination Port	4059
Layer 3 protocol	17
TOS byte	0
Ingress Interface	WPAN 0
Non-Key Fields	
Length	512
Packets count	23

Key Fields	Packet #2
Source IP	192.168.27.1
Destination IP	10.23.42.56
Source Port	45324
Destination Port	20000
Layer 3 protocol	6
TOS byte	0
Ingress Interface	Ethernet 0
Non-Key Fields	
Length	768
Packet counts	23

FNF Cache

Src. IP	Dest. IP	Src. Port	Dest. Port	Layer 3 Prot.	TOS Byte	Ingress Intf.	Timestamps	Length	Packets
2001:d8b:1:2::68	2001:d8b:3:4:250	27356	4059	17	0	Wpan 0		512	23
192.168.27.1	10.23.42.56	45324	20000	6	0	Ethernet 0		768	35

Figure 7-17 *Flexible NetFlow overview*

FNF Components

- FNF Flow monitor (NetFlow cache)-> describes the information stored in cache/ NetFlow cache. It contains two fields; key fields(used to create a flow, unique/flow record), and non-key fields(collected with the flow as attribute or characteristics of a flow) within the cache. One part of Flow monitor is called Flow Exporter, which contains information regarding the export of NetFlow information(destination address of NetFlow collector). It also includes various cache characteristics such as timers for exporting, the size of the cache, packet sampling rate (if required).
- FNF flow record-> it is a set of key and non-key NetFlow field values used to characterize flows in the NetFlow cache. Flow records may be pre-defined, customized, or user-defined. A typical pre-defined record aggregates flow data and allows users to target common applications for NetFlow. User-defined records allows selections of specific key or non-key fields in the flow record, and is the key to Flexible NetFlow, allowing wide range of information to be characterized and exported by NetFlow.
- FNF Exporter-> there are two primary methods for accessing NetFlow data: using the **show** commands at the command-line interface, and using an **application reporting tool.** It pushes information to the NetFlow reporting collector periodically.

- The flexible NetFlow Exporter allows the user to define where the export can be sent, the type of transport for the export, and the properties for the export. Multiple exporters can be configured per flow monitor.

❑ Flow export timers->timers indicate how often flows should be exported to the collection and reporting server

❑ NetFlow export format->indicates the type of flow reporting format

❑ NetFlow server for collection and reporting->is the destination of flow export. It is often done with the analytics tool that looks anomalies in the traffic patterns.

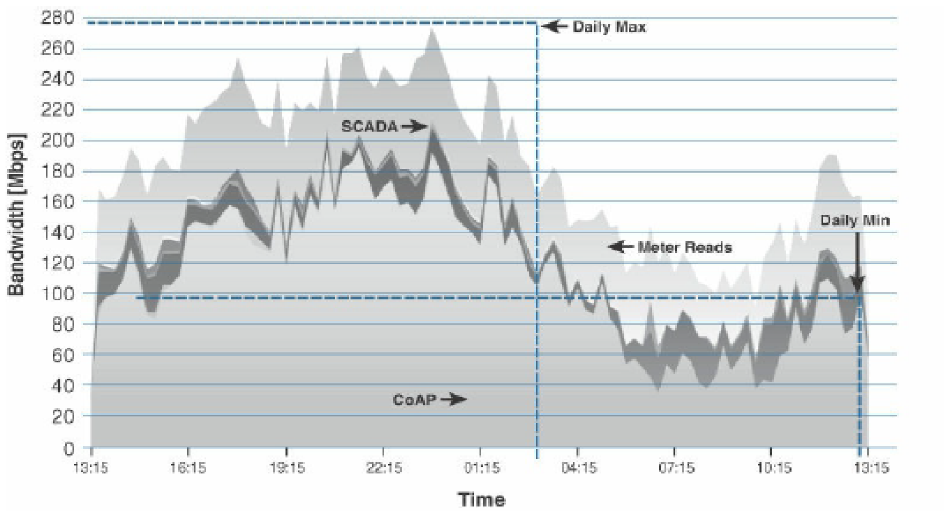


Figure 7-18 *FNF Report of Traffic on a Smart Grid FAN*

Key advantages of FNF are:

- Flexibility, scalability, and aggregation of flow data
- Ability to monitor a wide range of packet information and produce new information about network behaviour
- Enhanced network anomaly and security detection
- User-configurable flow information for performing customized traffic identification and ability to focus and monitor specific network behaviour
- Convergence of multiple accounting technologies into one accounting mechanism

Flexible NetFlow in Multiservice IoT Networks

- In the context of multiservice IoT networks, it is recommended that FNF be configured on the routers that aggregate connections from last mile's router. This gives a global view of all services flowing between the core network in the cloud and the IoT last-mile network.
- FNF can also be configured on the last-mile gateway or fog nodes for more granular visibility. However, flow analysis at the gateway is not possible with all IoT systems.

Challenges with deploying flow analytics tools in an IoT network include the following:

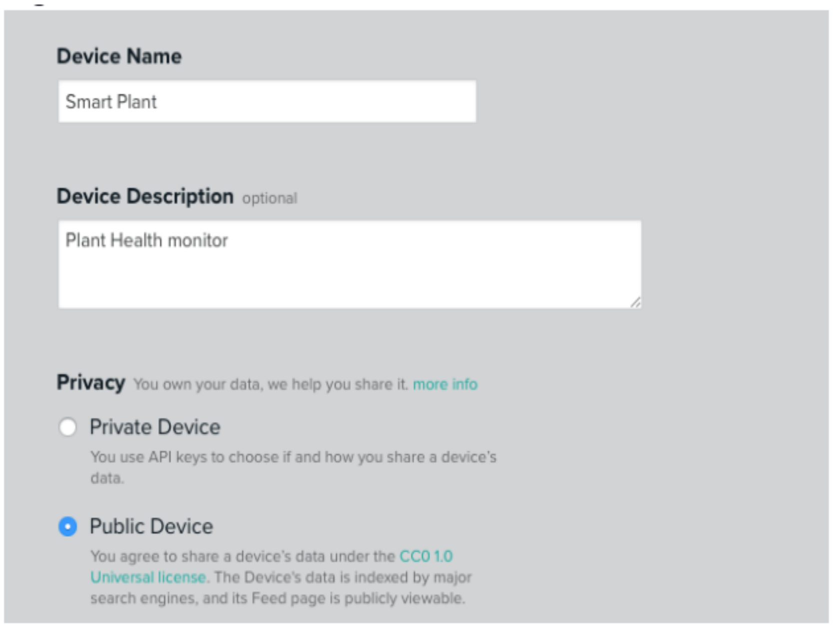
1. The distributed nature of fog and edge computing may mean that traffic flows are processed in places that might not support flow analytics, and thus visibility is lost
2. IPV4 and IPV6 native interfaces sometimes need to inspect inside VPN tunnels, which may impact the router's performance.
3. The added cost of increasing bandwidth due to additional network management traffic need to be reviewed, especially if the backhaul network uses cellular or satellite communications.

Xively Cloud

- It is a commercial platform-as-a-service that can be used for creating solutions for IoT. With Xively cloud, the IoT developers can focus on the front-end infrastructure and devices for IoT.
- Xively platform comprises of a message bus for real-time message management and routing, data services for time series archiving, directory services that provides a searchable directory of objects and business services for device provisioning and management.
- It supports, API over HTTP, sockets and MQTT for connecting IoT devices to the Xively cloud

Getting started with Xively

1. Add device to Xively account



Device Name

Smart Plant

Device Description optional

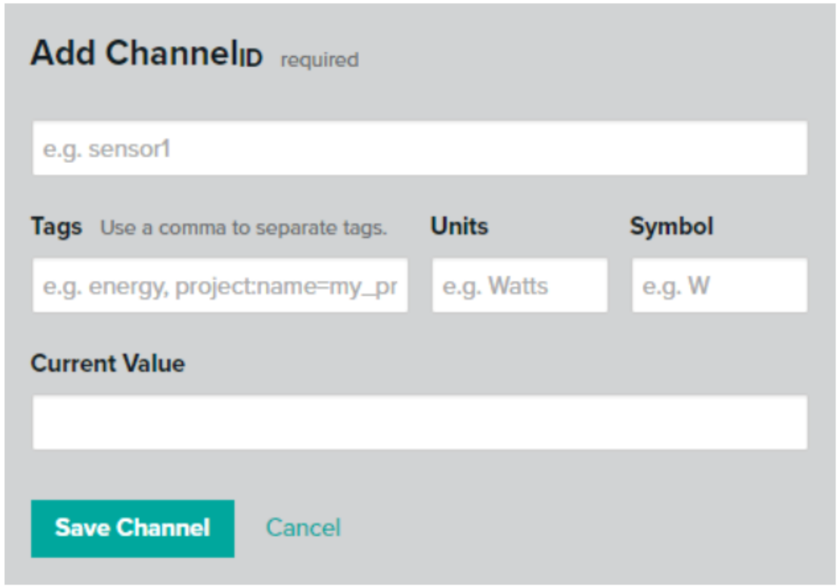
Plant Health monitor

Privacy You own your data, we help you share it. more info

Private Device
You use API keys to choose if and how you share a device's data.

[✓ Public Device]
You agree to share a device's data under the CC0 1.0 Universal license. The Device's data is indexed by major search engines, and its Feed page is publicly viewable.

2. Define channels for account



Add ChannelID required

e.g. sensor1

Tags Use a comma to separate tags. | Units | Symbol

e.g. energy, project:name=my_pr | e.g. Watts | e.g. W

Current Value

3. Write the code and run

https://dzone.com/articles/how-to-use-xively-platform-in-iot-project

Python Web application Frameworks

To build IoT applications that are backed by either Xively cloud or any other data collection systems, we need web application framework. Some of them are presented below:

Django->

- It is an open source web application framework for developing web application in python.
- It is based on the well-known Model-Template-View architecture and provides a separation of the data model from the business rules and the user interface.
- Django provides an unified API to a database backend. Therefore, web applications built with Django can work with different databases without any code-base changes.
- Django consists of an object-relational mapper, a web templating system and a regular-expression-based URL dispatcher.

Django Architecture

Django is a Model-Template-View (MTV) framework wherein the roles model, template and view, respectively are:

- Model-> it acts as a definition of some stored data and handles the interaction with the databases. In web application, the data can be stored in a relational, non-relational database, in XML file etc. A Django model is a python class that outlines the variables and methods for a particular type of data.
- Template-> template is simply an HTML page with a few extra place holders. Django's template language can be used to create various forms of text files (XML, email, CSS, Javascript, CSV etc).
- View-> view ties the model to the template. The view is where the code is actually written to generate the webpages. View determines what data need to be displayed, retrieves the data from the database, and passes the data to the template.

Getting started with Django

- **Install Django:** https://docs.djangoproject.com/en/4.1/intro/install/
- **Creating a Django Project and App:** https://docs.djangoproject.com/en/4.1/intro/tutorial01/

Django comes with a built-in lightweight web server can be viewed at the URL: http://localhost:8000

- **Configuring a database:** it supports MySQL, PostgreSQL, Oracle and SQLite3(relational database), and MongoDB (non-relational database): install MySQL
- **Defining a model:** it acts as a definition of the data in the database (optional)
- **Defining a view:** Django views are Python functions that takes http requests and returns http response, like HTML documents.

https://www.w3schools.com/django/django_views.php

- **Defining URL patterns:** URL patterns are a way of mapping the URLs to the views that should handle the URL requests

https://www.w3schools.com/django/django_urls.php

Example: https://realpython.com/get-started-with-django-1/

AWS(Amazon Web Services)

- **Amazon EC2 (Elastic Compute Cloud)->** it is an Infrastructure-as-a-service (Iaas) provided by Amazon and pay-as-you-go compute capacity in the cloud. It is a webservice that provides computing capacity in the form of virtual machines that is present in Amazon's cloud computing platform. It supports auto-scaling capacity.
- **Amazon S3 (Simple Storage Service)->** it is an online cloud-based data storage infrastructure for storing and retrieving a large amount of data.
- **Amazon RDS->** it is a web service that allows to create instances of MySQL, Oracle, or Microsoft SQL server in the cloud. Using RDS, developers can easily set up, operate, and scale a relational database in the cloud.
- **Amazon DynamoDB->** it is a fully managed, scalable, high performance No-SQL database service. DynamoDB can serve as a scalable datastore for IoT systems. With DynamoDB, any amount of data can be stored and can be served with any level of requests for the data.

https://aws.amazon.com/console/

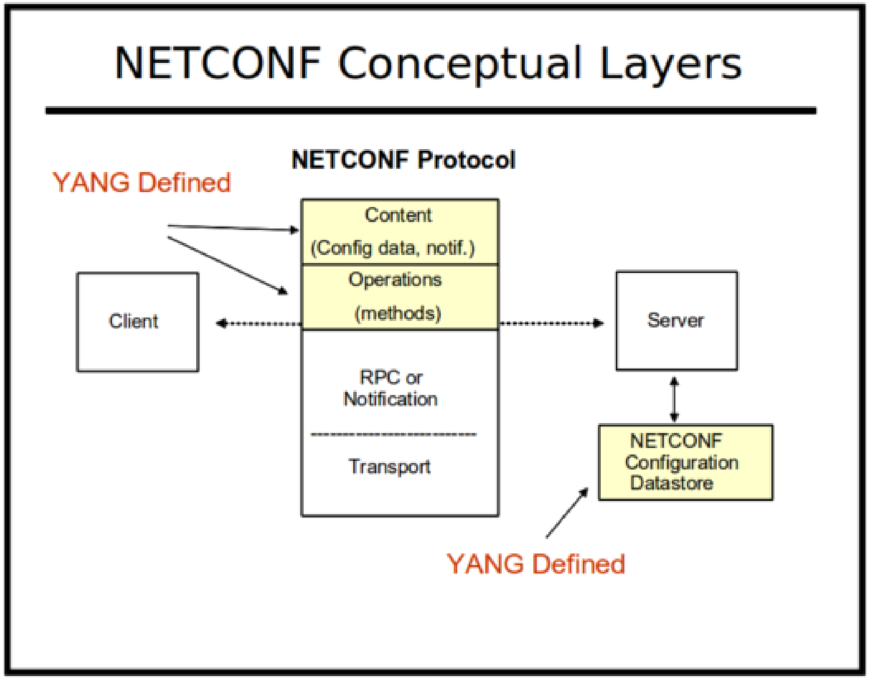
SkyNet IoT Messaging Platform

- SkyNet is an open source instant messaging platform for IoT. It supports both HTTP REST and real-time web sockets
- It allows to register nodes (devices) in a network. A device can be sensor, cloud resources, drones, smart home devices etc. Each device is having a UUID and a secret token.
- Devices or client applications can subscribe to other devices and send/ receive messages.

https://support.skynetlabs.com/

NETCONF

- Network Configuration Protocol (NETCONF) is a session-based network management protocol.
- NETCONF allows retrieving state or configuration data and manipulating configuration data on network devices.



- Here, server is the management system, and client is the device
- It supports TLS, SSH protocols for transport security, as well as provides end-to-end connectivity and ensure reliable delivery of messages.
- NETCONF uses XML-encoded Remote Procedure Calls(RPCs) for framing request and response messages. The RPC layer provides mechanism for encoding of RPC calls and notifications.
- NETCONF provides various operations to retrieve and edit configuration data from network devices. Some commonly used NETCONF operations are; connect, get, get-config, edit-config, copy-config, lock, unlock, commit, close-session, kill-session etc.
- The content layer consists of configuration and state data which is XML-encoded the schema of the configuration and state data is defined in a data modelling language called YANG.

- For managing a network device the client establishes a NETCONF session with the server.
- When session is established the client and server exchange “hello” messages which contains information of their capabilities. The client can send multiple requests to the server for retrieving or editing the configuration data.
- NETCONF defines one or more configuration data stores. A configuration store contains all the configuration information to bring the device from its initial state to the operational state.
- By default a <running> configuration store is present. Additional configuration datastores such as <startup> and <candidate> can be defined in the capabilities.
- NETCONF is a connection oriented protocol and NETCONF connections persists between protocol operations. For authentication, data integrity, and confidentiality, NETCONF depends on transport protocol SSH or TLS.

YANG

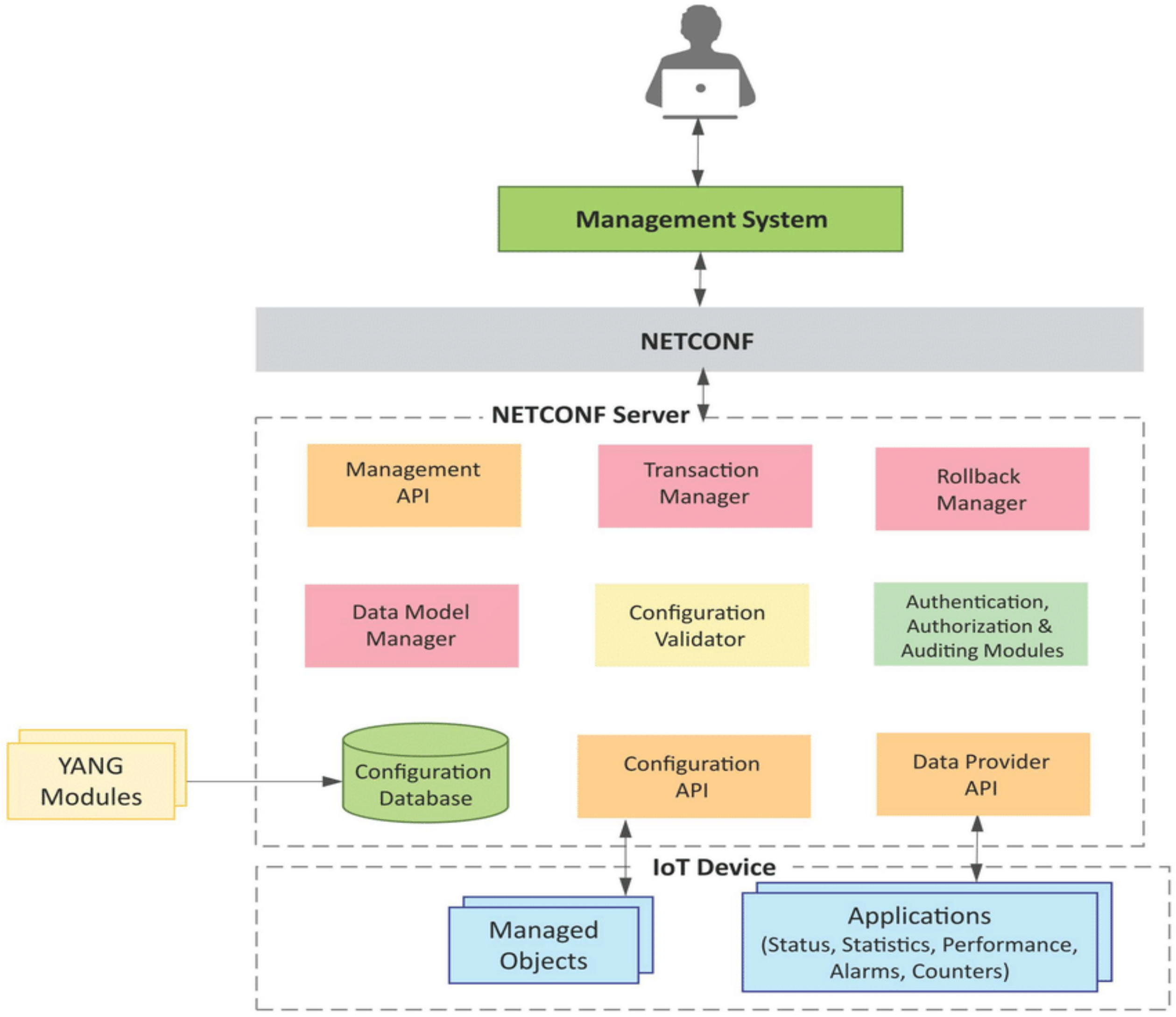
- YANG is a data modelling language used to model configuration and state data manipulated by the NETCONF protocol.
- YANG modules contain the definitions of the configuration data, state data, RPC calls that can be issued and the format of the notifications.
- YANG modules defines the data exchanged between the NETCONF client and server.
- A module comprises of a number of “leaf nodes” which are organized into a hierarchical tree structure. The leaf nodes are organized using “container” or “list” constructs.
- A YANG module can import definitions from other modules.
- Constraints can be defined on the data nodes, e.g., allowed values.
- YANG can model both configuration data and state data using the “config” statement.
- YANG defines 4 types of nodes for data modelling:

1. Leaf Nodes-> contains simple data structures such as integer or string
2. Leaf-List Nodes-> is a sequence of leaf nodes with exactly one value of a particular type per leaf
3. Container Nodes-> used to group related nodes in a subtree. A container may contain any number of child nodes of any type.
4. List Nodes-> defines a sequence of list entries. Each entry is like a structure or a record instance, and is uniquely identified by the values of its key leaves. A list can contain multiple leaves and may contain any number of child nodes of any type.

https://www.cisco.com/c/en/us/support/docs/storage-networking/management/200933-YANG-NETCONF-Configuration-Validation.html

IoT system management with NETCONF-YANG

- Management System
- Management API
- Transactional Manager
- Rollback Manager
- Data Model Manager
- Configuration Validator
- Configuration database
- Configuration API
- Data Provider API

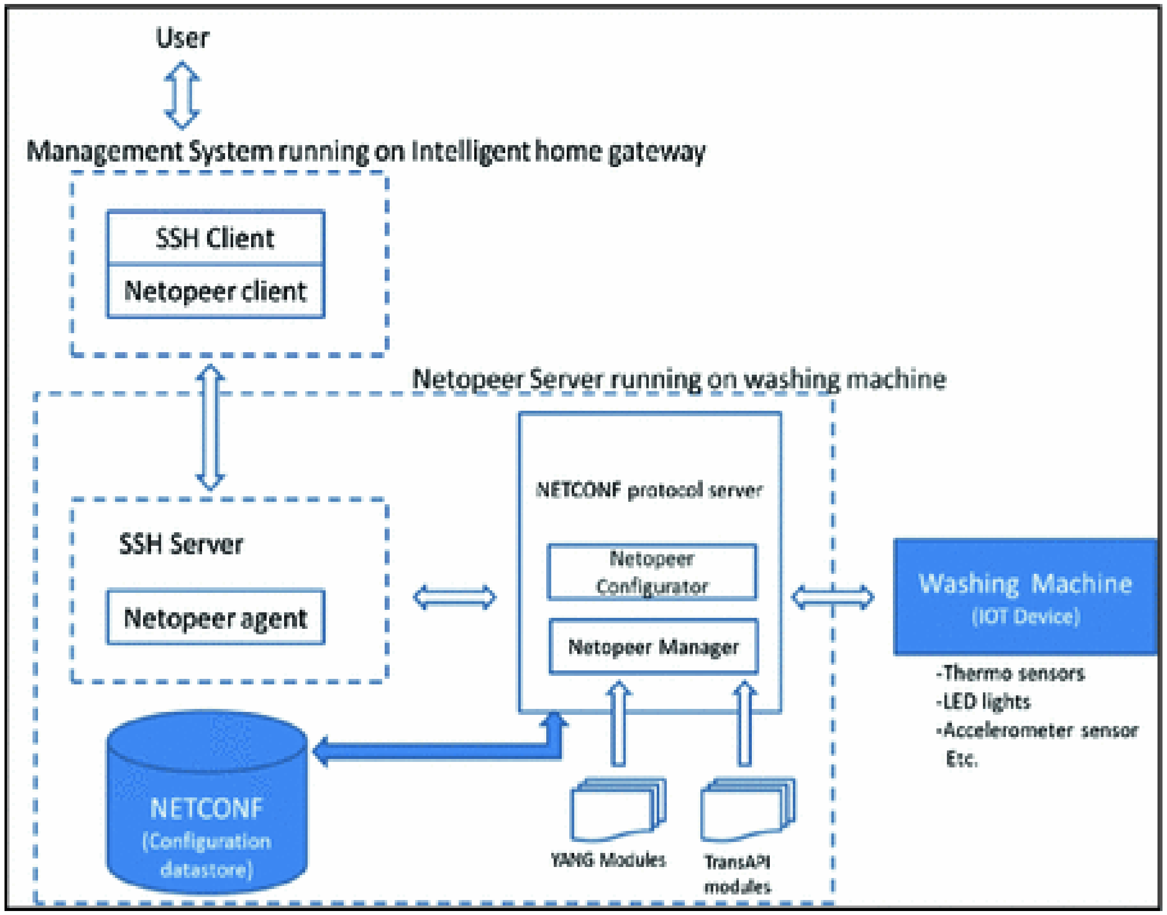


- Management Systems-> the operator uses a management system to send NETCONF messages to configure the IoT device and receives state information and notifications from the device as NETCONF messages.
- Management API-> allows management applications to start NETCONF sessions, read and write configuration data, retrieve configurations.
- Transaction Manager-> executes all the NETCONF transactions and ensures that the ACID properties hold true.
- Rollback Manager-> is responsible for generating all the transactions necessary to rollback a current configuration to its original state.
- Data Model Manager-> keeps track of all the YANG data models and the corresponding managed objects. Also keeps track of the applications which provide data for each part of a data model.
- Configuration Validator-> checks if the resulting configuration after applying a transaction would be a valid configuration.
- Configuration Database-> contains both the configurational and operational data.
- Configuration API-> using this the applications on the IoT device can read configuration data from the configuration datastore and write operational data to the operational datastore.
- Data Provider API-> applications on the IoT device can register for callbacks for various events using the Data Provider API. Through the Data Provider API, the applications can report statistics and operational data.

Netopeer:

- Netopeer is set of open source NETCONF tools built on the Libnetconf library

- Netopeer-server
- Netopeer-agent
- Netopeer-cli
- Netopeer-manager
- Netopeer-configurator



- Netopper-server-> is a NETCONF protocol server that runs on the managed device. It provides an environment for configuring the device using NETCONF RPC operations and also retrieving the state data from the device.
- Netopper-agent-> is the NETCONF protocol agent running as a SSH/TLS subsystem. It accepts incoming NETCONF connection and passes the NETCONF RPC operations received from the NETCONF client to the Netopper-server.
- Netopper-cli-> is a NETCONF client that provides a command line interface for interacting with the Netopper-server. The operator can use the Netopper-cli to send NETCONF RPC operations for configuring the device and retrieving the state information.
- Netopper-manager-> allows managing the YANG and Libnetconf Transaction API (TransAPI) modules on the Netopper-server. With Netopper-manager modules can be loaded or removed from the server.
- Netopper-configurator-> is a tool that can be used to configure the Netopper-server.

Steps for IoT device management with NETCONF-YANG:

1. Create a YANG model of the system that defines the configuration and state of the system
2. Compile the YANG model with the “lnctool” which comes with Libnetconf.
3. Fill in the IoT device management code in the TransAPI module(callbacks C file). This file includes configuration callbacks, RPC callbacks and state callbacks.
4. Build the callbacks C file to generate the library file (.so).
5. Load the YANG module and the TransAPI module into the Netopeer server using the Netopeer manager tool.
6. The operator can now connect from the management system to the Netopeer server using the Netopeer CLI.
7. Operator can issue NETCONF commands from the Netopeer CLI. Commands can be issued to change the configuration data, get operational data or execute an RPC on the IoT device.

Examples:

https://community.cisco.com/t5/networking-blogs/getting-started-with-netconf-yang-part-1/ba-p/3661241

https://networkop.co.uk/blog/2017/01/25/netconf-intro/