1. Discuss the diff steps & phases of octave allego methodology :

Formal Risk Analysis Structures: OCTAVE and FAIR

Within the industrial environment, there are a number of standards, guidelines, and best practices available to help understand risk and how to mitigate it. IEC 62443 is the most commonly used standard globally across industrial verticals. It consists of a number of parts, including 62443-3-2 for risk assessments, and 62443-3-3 for foundational requirements used to secure the industrial environment from a networking and communications perspective. Also, ISO 27001 is widely used for organizational people, process, and information security management. In addition, the National Institute of Standards and Technology (NIST) provides a series of documents for critical infrastructure, such as the NIST Cybersecurity Framework (CSF). In the utilities domain, the North American Electric Reliability Corporation's (NERC's) Critical Infrastructure Protection (CIP) has legally binding guidelines for North American utilities, and IEC 62351 is the cybersecurity standard for power utilities.

The key for any industrial environment is that it needs to address security holistically and not just focus on technology. It must include people and processes, and it should include all the vendor ecosystem components that make up a control system.

In this section, we present a brief review of two such risk assessment frameworks:

- OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) from the Software Engineering Institute at Carnegie Mellon University
- FAIR (Factor Analysis of Information Risk) from The Open Group

These two systems work toward establishing a more secure environment but with two different approaches and sets of priorities. Knowledge of the environment is key to determining security risks and plays a key role in driving priorities.

OCTAVE

OCTAVE has undergone multiple iterations. The version this section focuses on is OCTAVE Allegro, which is intended to be a lightweight and less burdensome process to implement. Allegro assumes that a robust security team is not on standby or immediately

at the ready to initiate a comprehensive security review. This approach and the assumptions it makes are quite appropriate, given that many operational technology areas are similarly lacking in security-focused human assets. Figure 8-5 illustrates the OCTAVE Allegro steps and phases.

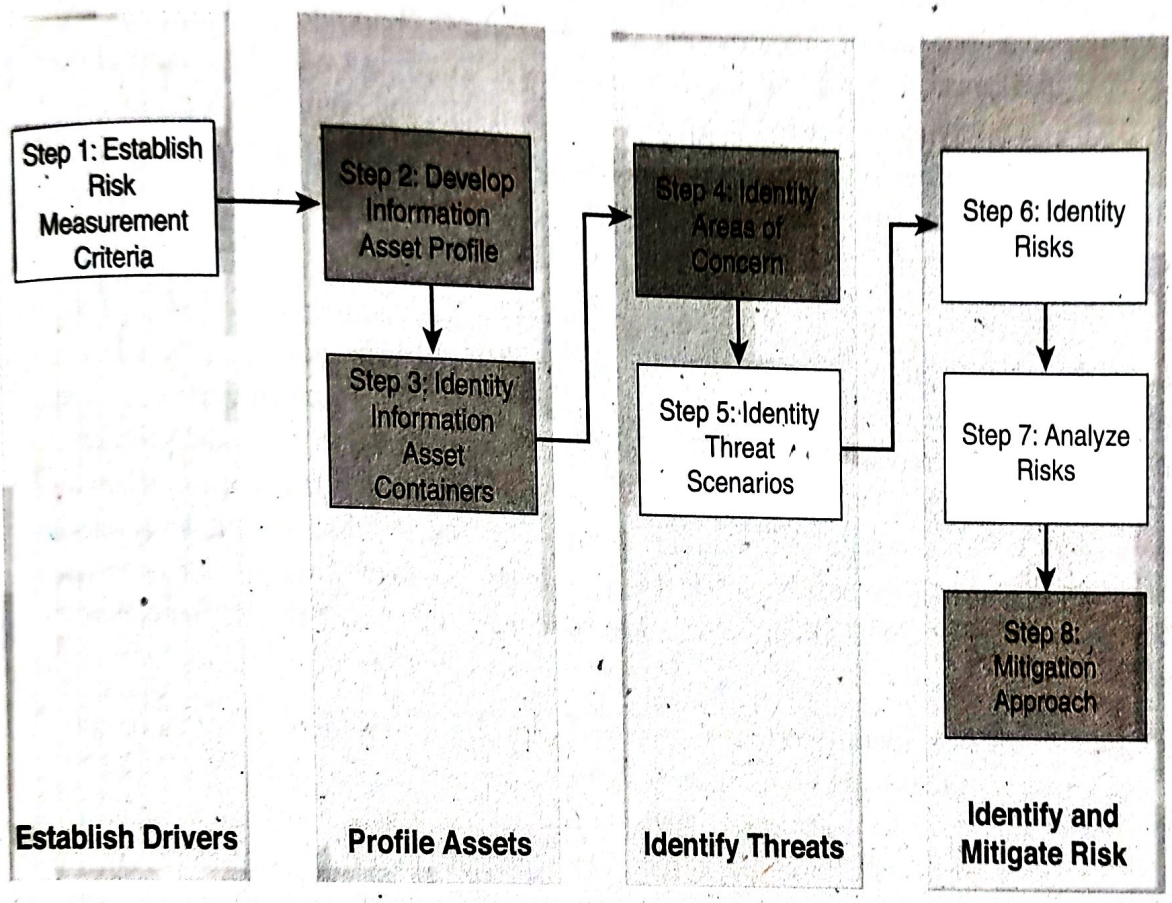

Figure 8-5 *OCTAVE Allegro Steps and Phases (see https://blog.compass-security.com/2013/04/lean-risk-assessment-based-on-octave-allegro/).*

The first step of the OCTAVE Allegro methodology is to establish a risk measurement criterion. OCTAVE provides a fairly simple means of doing this with an emphasis on impact, value, and measurement. The point of having a risk measurement criterion is that at any point in the later stages, prioritization can take place against the reference model. (While OCTAVE has more details to contribute, we suggest using the FAIR model, described next, for risk assessment.)

The second step is to develop an information asset profile. This profile is populated with assets, a prioritization of assets, attributes associated with each asset, including owners, custodians, people, explicit security requirements, and technology assets. It is important to stress the importance of process. Certainly, the need to protect information does not disappear, but operational safety and continuity are more critical.

Within this asset profile, process are multiple substages that complete the definition of the assets. Some of these are simply survey and reporting activities, such as identifying the asset and attributes associated with it, such as its owners, custodians, human actors with which it interacts, and the composition of its technology assets. There are, however, judgment-based attributes such as prioritization. Rather than simply assigning an

arbitrary ranking, the system calls for a justification of the prioritization. With an understanding of the asset attributes, particularly the technical components, appropriate threat mitigation methods can be applied. With the application of risk assessment, the level of security investment can be aligned with that individual asset.

The third step is to identify information asset containers. Roughly speaking, this is the range of transports and possible locations where the information might reside. This references the compute elements and the networks by which they communicate. However, it can also mean physical manifestations such as hard copy documents or even the people who know the information. Note that the operable target here is information, which includes data from which the information is derived.

In OCTAVE, the emphasis is on the container level rather than the asset level. The value is to reduce potential inhibitors within the container for information operation. In the OT world, the emphasis is on reducing potential inhibitors in the containerized operational space. If there is some attribute of the information that is endemic to it, then the entire container operates with that attribute because the information is the defining element. In some cases this may not be true, even in IT environments. Discrete atomic-level data may become actionable information only if it is seen in the context of the rest of the data. Similarly, operational data taken without knowledge of the rest of the elements may not be of particular value either.

The fourth step is to identify areas of concern. At this point, we depart from a data flow, touch, and attribute focus to one where judgments are made through a mapping of security-related attributes to more business-focused use cases. At this stage, the analyst looks to risk profiles and delves into the previously mentioned risk analysis. It is no longer just facts, but there is also an element of creativity that can factor into the evaluation. History both within and outside the organization can contribute. References to similar operational use cases and incidents of security failures are reasonable associations.

Closely related is the fifth step, where threat scenarios are identified. Threats are broadly (and properly) identified as potential undesirable events. This definition means that results from both malevolent and accidental causes are viable threats. In the context of operational focus, this is a valuable consideration. It is at this point that an explicit identification of actors, motives, and outcomes occurs. These scenarios are described in threat trees to trace the path to undesired outcomes, which, in turn, can be associated with risk metrics.

At the sixth step risks are identified. Within OCTAVE, risk is the possibility of an undesired outcome. This is extended to focus on how the organization is impacted. For more focused analysis, this can be localized, but the potential impact to the organization could extend outside the boundaries of the operation.

The seventh step is risk analysis, with the effort placed on qualitative evaluation of the impacts of the risk. Here the risk measurement criteria defined in the first step are explicitly brought into the process.

Finally, mitigation is applied at the eighth step. There are three outputs or decisions to be taken at this stage. One may be to accept a risk and do nothing, other than document the

situation, potential outcomes, and reasons for accepting the risk. The second is to mitigate the risk with whatever control effort is required. By walking back through the threat scenarios to asset profiles, a pairing of compensating controls to mitigate those threat/risk pairings should be discoverable and then implemented. The final possible action is to defer a decision, meaning risk is neither accepted nor mitigated. This may imply further research or activity, but it is not required by the process.

OCTAVE is a balanced information-focused process. What it offers in terms of discipline and largely unconstrained breadth, however, is offset by its lack of security specificity. There is an assumption that beyond these steps are seemingly means of identifying specific mitigations that can be mapped to the threats and risks exposed during the analysis process.

2. Explain purdue model for control hierarchy?

The Purdue Model for Control Hierarchy

Regardless of where a security threat arises, it must be consistently and unequivocally treated. IT information is typically used to make business decisions, such as those in process optimization, whereas OT information is instead characteristically leveraged to make physical decisions, such as closing a valve, increasing pressure, and so on. Thus, the operational domain must also address physical safety and environmental factors as part of its security strategy—and this is not normally associated with the IT domain. Organizationally, IT and OT teams and tools have been historically separate, but this has begun to change, and they have started to converge, leading to more traditionally IT-centric solutions being introduced to support operational activities. For example, systems such as firewalls and intrusion prevention systems (IPS) are being used in IoT networks.

As the borders between traditionally separate OT and IT domains blur, they must align strategies and work more closely together to ensure end-to-end security. The types of devices that are found in industrial OT environments are typically much more highly optimized for tasks and industrial protocol-specific operation than their IT counterparts. Furthermore, their operational profile differs as well.

Industrial environments consist of both operational and enterprise domains. To understand the security and networking requirements for a control system, the use of a logical framework to describe the basic composition and function is needed. The Purdue Model for Control Hierarchy, introduced in Chapter 2, is the most widely used framework across industrial environments globally and is used in manufacturing, oil and gas, and many other industries. It segments devices and equipment by hierarchical function levels and areas and has been incorporated into the ISA99/IEC 62443 security standard, as shown in Figure 8-3. For additional detail on how the Purdue Model for Control Hierarchy is applied to the manufacturing and oil and gas industries, see Chapter 9, "Manufacturing," and Chapter 10, "Oil and Gas."

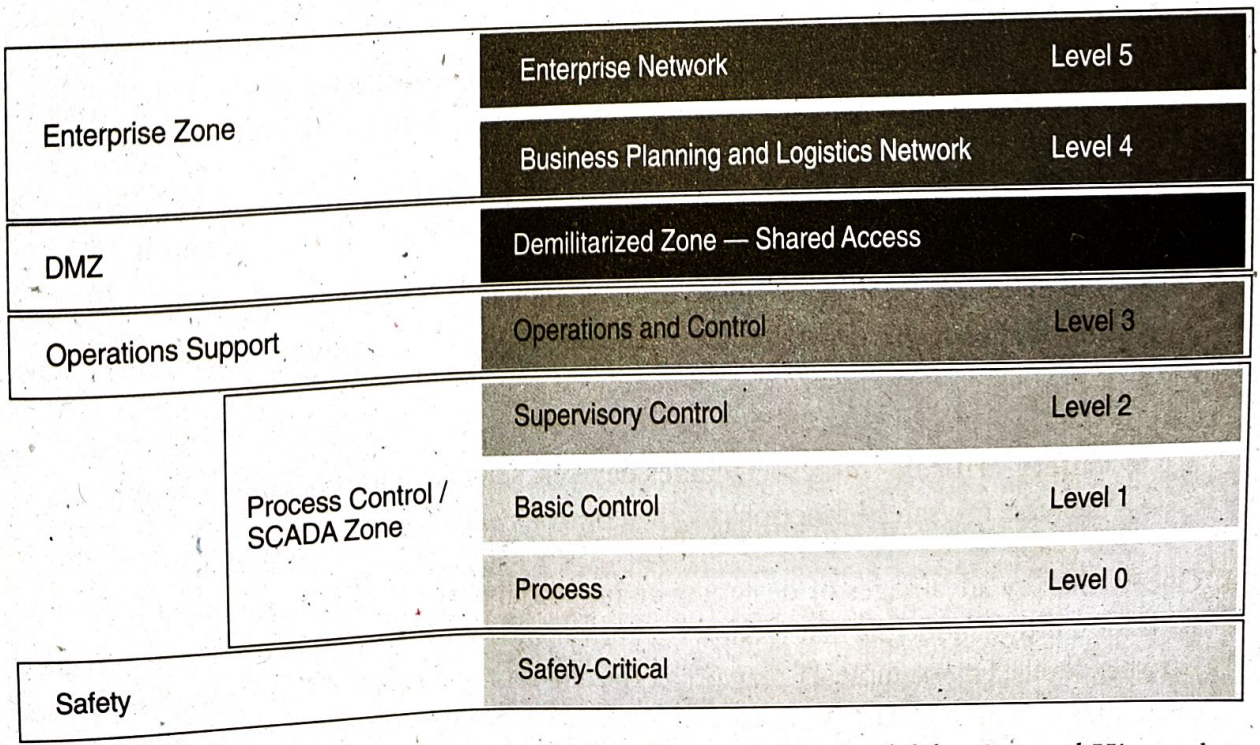

Figure 8-3 *The Logical Framework Based on the Purdue Model for Control Hierarchy*

This model identifies levels of operations and defines each level. The enterprise and operational domains are separated into different zones and kept in strict isolation via an industrial demilitarized zone (DMZ):

- Enterprise zone
  - **Level 5: Enterprise network:** Corporate-level applications such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), document management, and services such as Internet access and VPN entry from the outside world exist at this level.
  - **Level 4: Business planning and logistics network:** The IT services exist at this level and may include scheduling systems, material flow applications, optimization and planning systems, and local IT services such as phone, email, printing, and security monitoring.
- Industrial demilitarized zone
  - **DMZ:** The DMZ provides a buffer zone where services and data can be shared between the operational and enterprise zones. It also allows for easy segmentation of organizational control. By default, no traffic should traverse the DMZ; everything should originate from or terminate on this area.
- Operational zone
  - **Level 3: Operations and control:** This level includes the functions involved in managing the workflows to produce the desired end products and for monitoring and controlling the entire operational system. This could include production scheduling, reliability assurance, systemwide control optimization, security management, network management, and potentially other required IT services, such as DHCP, DNS, and timing.
  - **Level 2: Supervisory control:** This level includes zone control rooms, controller status, control system network/application administration, and other control-related applications, such as human-machine interface (HMI) and historian.
  - **Level 1: Basic control:** At this level, controllers and IEDs, dedicated HMIs, and other applications may talk to each other to run part or all of the control function.
  - **Level 0: Process:** This is where devices such as sensors and actuators and machines such as drives, motors, and robots communicate with controllers or IEDs.
- Safety zone
  - **Safety-critical:** This level includes devices, sensors, and other equipment used to manage the safety functions of the control system.

3. Explain in detail core functions of edge analytics?

Edge Analytics Core Functions

To perform analytics at the edge, data needs to be viewed as real-time flows. Whereas big data analytics is focused on large quantities of data at rest, edge analytics continually processes streaming flows of data in motion. Streaming analytics at the edge can be broken down into three simple stages:

- **Raw input data:** This is the raw data coming from the sensors into the analytics processing unit.
- **Analytics processing unit (APU):** The APU filters and combines data streams (or separates the streams, as necessary), organizes them by time windows, and performs various analytical functions. It is at this point that the results may be acted on by micro services running in the APU.
- **Output streams:** The data that is output is organized into insightful streams and is used to influence the behavior of smart objects, and passed on for storage and further processing in the cloud. Communication with the cloud often happens through a standard publisher/subscriber messaging protocol, such as MQTT.

Figure 7-12 illustrates the stages of data processing in an edge APU.

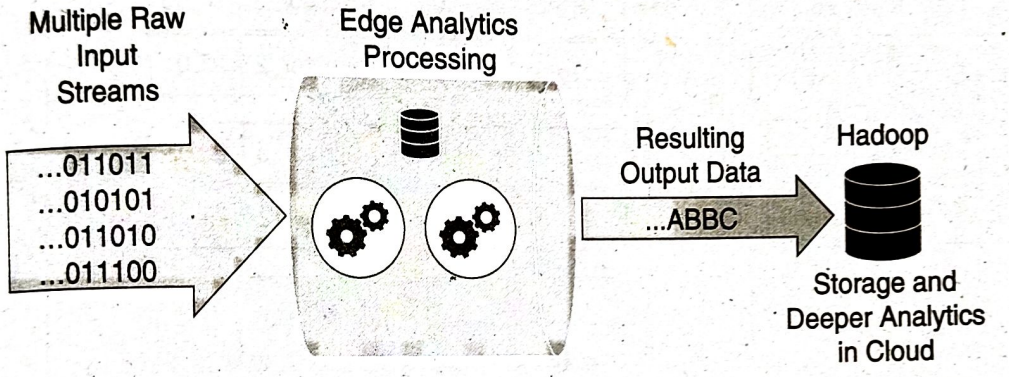

Figure 7-12 *Edge Analytics Processing Unit*

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

- **Filter:** The streaming data generated by IoT endpoints is likely to be very large, and most of it is irrelevant. For example, a sensor may simply poll on a regular basis to confirm that it is still reachable. This information is not really relevant and can be mostly ignored. The filtering function identifies the information that is considered important.
- **Transform:** In the data warehousing world, Extract, Transform, and Load (ETL) operations are used to manipulate the data structure into a form that can be used for other purposes. Analogous to data warehouse ETL operations, in streaming analytics, once the data is filtered, it needs to be formatted for processing.
- **Time:** As the real-time streaming data flows, a timing context needs to be established. This could be to correlated average temperature readings from sensors on a minute-by-minute basis. For example, Figure 7-13 shows an APU that takes input data from multiple sensors reporting temperature fluctuations. In this case, the APU is programmed to report the average temperature every minute from the sensors, based on an average of the past two minutes. (An example where this may be used is in real-time monitoring of food in a grocery store, where rolling averages of the temperature in open-air refrigeration units needs to be monitored to ensure the safety of

the food.) Note that on the left side is the cleaned stream data. This data is presented as streams to the analytics engine (note the syntax at the bottom right of the figure) that establishes the time window and calculates the average temperature over the past two minutes. The results are reported on a per-minute basis (on the right side of the figure).

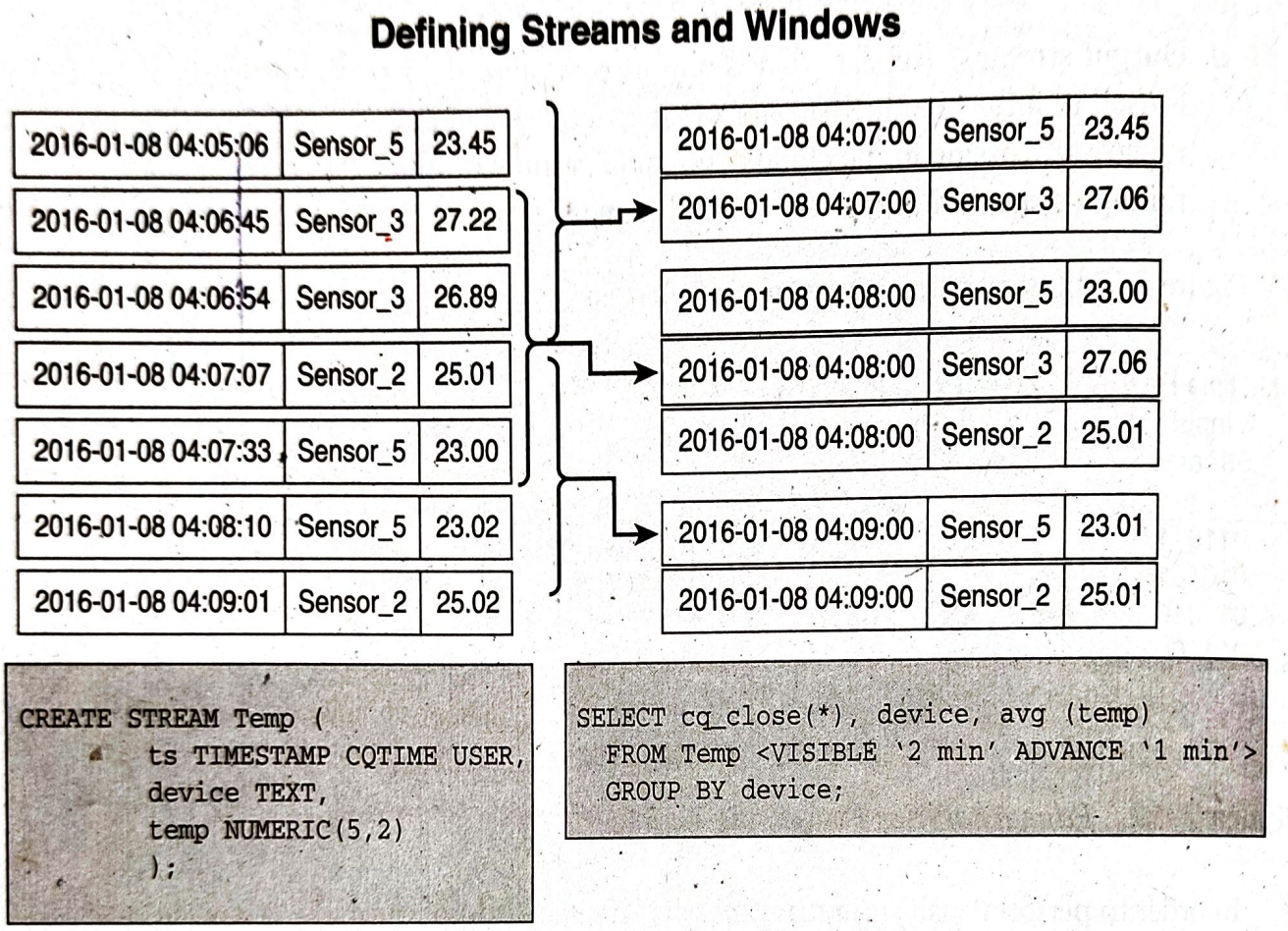

Figure 7-13 *Example: Establishing a Time Window for Analytics of Average Temperature from Sensors*

- **Correlate:** Streaming data analytics becomes most useful when multiple data streams are combined from different types of sensors. For example, in a hospital, several vital signs are measured for patients, including body temperature, blood pressure, heart rate, and respiratory rate. These different types of data come from different instruments, but when this data is combined and analyzed, it provides an invaluable picture of the health of the patient at any given time.[4] However, correlation goes beyond just combining real-time data streams. Another key aspect is combining and correlating real-time measurements with preexisting, or historical, data. For example, historical data may include the patient's past medical history, such as blood test results. Combining historical data gives the live streaming data a powerful context and promotes more insights into the current condition of the patient (see Figure 7-14).

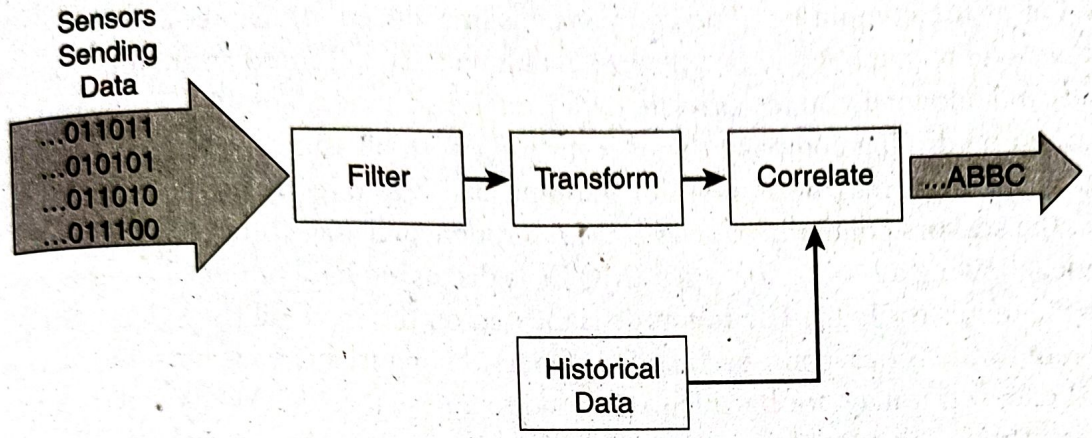

Figure 7-14 *Correlating Data Streams with Historical Data*

- **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. For example, say that the APU has been collecting the patient's vitals for some time and has gained an understanding of the expected patterns for each variable being monitored. If an unexpected event arises, such as a sudden change in heart rate or respiration, the pattern matching operator recognizes this as out of the ordinary and can take certain actions, such as generating an alarm to the nursing staff. The patterns can be simple relationships, or they may be complex, based on the criteria defined by the application. Machine learning may be leveraged to identify these patterns.

- **Improve business intelligence:** Ultimately, the value of edge analytics is in the improvements to business intelligence that were not previously available. For example, conducting edge analytics on patients in a hospital allows staff to respond more quickly to the patient's changing needs and also reduces the volume of unstructured (and not always useful) data sent to the cloud. Over time, the resulting changes in business logic can produce improvements in basic operations, bringing in higher levels of care as well as better efficiencies for the hospital.

4. Describe the components of FNF?

FNF Components

FNF has the following main components, as shown in Figure 7-17:

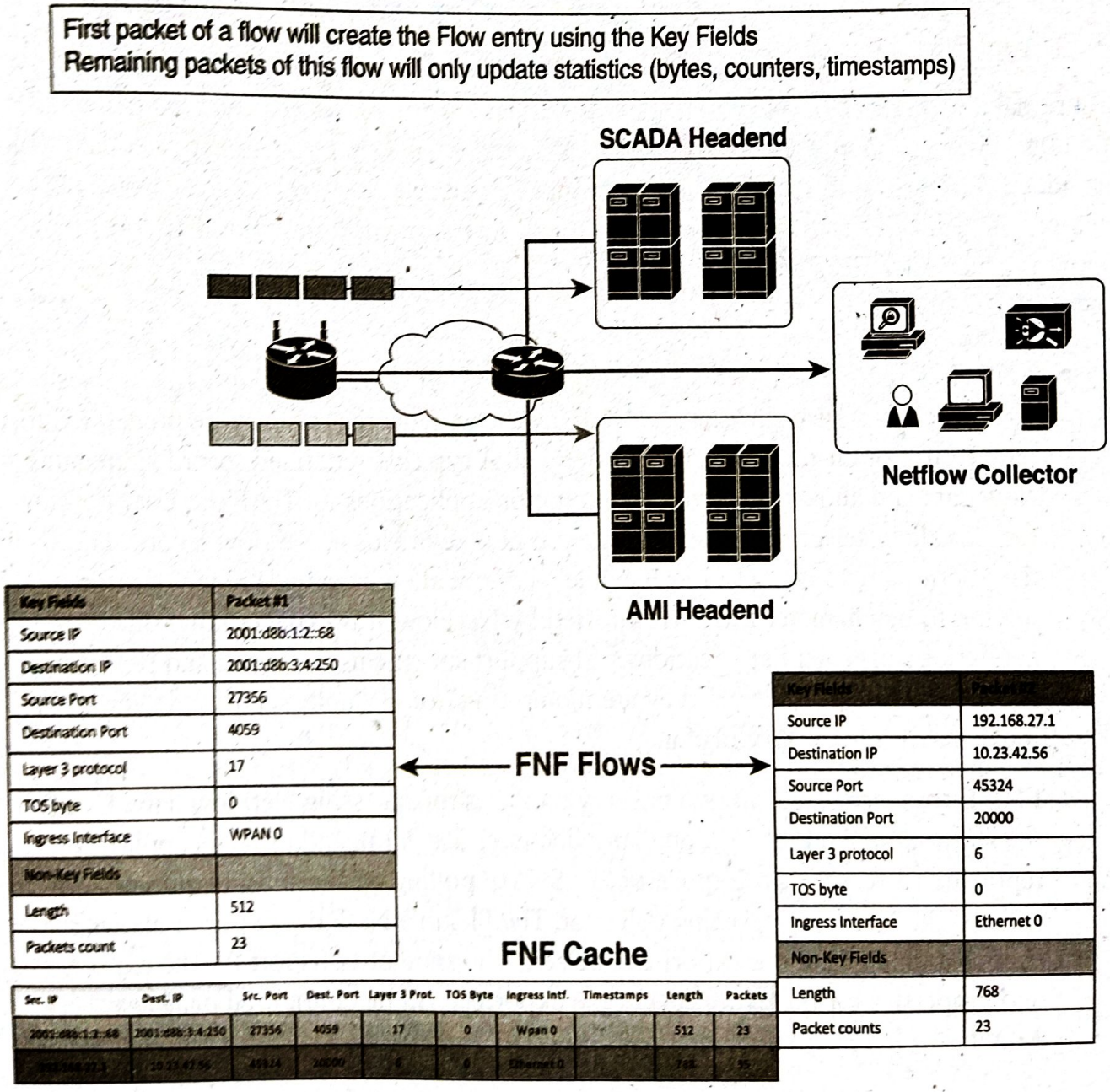

Figure 7-17 *Flexible NetFlow overview*

- **FNF Flow Monitor (NetFlow cache):** The FNF Flow Monitor describes the NetFlow cache or information stored in the cache. The Flow Monitor contains the flow record definitions with key fields (used to create a flow, unique per flow record: match statement) and non-key fields (collected with the flow as attributes or characteristics of a flow) within the cache. Also, part of the Flow Monitor is the Flow Exporter, which contains information about the export of NetFlow information, including the destination address of the NetFlow collector. The Flow Monitor includes various cache characteristics, including timers for exporting, the size of the cache, and, if required, the packet sampling rate.

Note Each packet that is forwarded within a router or switch is examined for a set of IP packet attributes. These attributes are the IP packet identity, or *key fields*, for the flow and determine whether the packet information is unique or similar to other packets. If packet key fields are unique, a new entry in the flow record is created. The first packet of a flow creates the flow entry, using the key fields. Remaining packets of this flow only update statistics (bytes, counters, timestamps). This methodology of flow characterization is scalable because a large amount of network information is condensed into a database of NetFlow information called the *NetFlow cache*.

Additional information (non-key fields) can be added to the Flow Record and exported. The non-key fields are not used to create or characterize the flows but are exported and just added to the flow. If a field is non-key, normally only the first packet of the flow is used for the value in this field. Examples include flow timestamps, next-hop IP addresses, subnet masks, and TCP flags.

- **FNF flow record:** A flow record is a set of key and non-key NetFlow field values used to characterize flows in the NetFlow cache. Flow records may be predefined for ease of use or customized and user defined. A typical predefined record aggregates flow data and allows users to target common applications for NetFlow. User-defined records allow selections of specific key or non-key fields in the flow record. The user-defined field is the key to Flexible NetFlow, allowing a wide range of information to be characterized and exported by NetFlow. It is expected that different network management applications will support specific user-defined and predefined flow records based on what they are monitoring (for example, security detection, traffic analysis, capacity planning).

- **FNF Exporter:** There are two primary methods for accessing NetFlow data: Using the **show** commands at the command-line interface (CLI), and using an application reporting tool. NetFlow Export, unlike SNMP polling, pushes information periodically to the NetFlow reporting collector. The Flexible NetFlow Exporter allows the user to define where the export can be sent, the type of transport for the export, and 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:** This simply indicates the type of flow reporting format.
- **NetFlow server for collection and reporting:** This is the destination of the flow export. It is often done with an analytics tool that looks for anomalies in the traffic patterns.

Figure 7-18 illustrates the analysis reported from the FNF records on a smart grid FAN. In this example, the FNF collector is able to see the patterns of traffic for various applications as well as management traffic on the FAN.

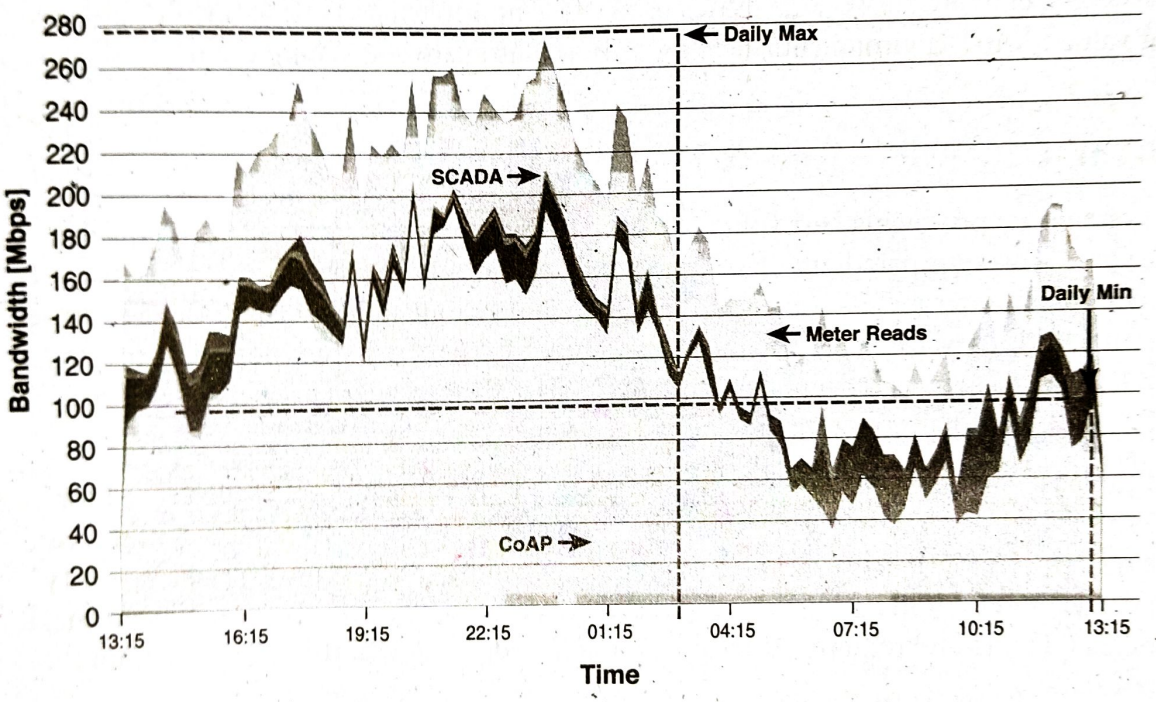

Figure 7-18 *FNF Report of Traffic on a Smart Grid FAN*

5. Explain neural nlw with example?

Neural Networks

Processing multiple dimensions requires a lot of computing power. It is also difficult to determine what parameters to input and what combined variations should raise red flags. Similarly, supervised learning is efficient only with a large training set; larger training sets usually lead to higher accuracy in the prediction. This requirement is partly what made ML fade away somewhat in the 1980s and 1990s. Training the machines was often deemed too expensive and complicated.

Since the 2000s, cheaper computing power along with access to very large data sets (shared over the Internet) rejuvenated the possibilities of ML. At the same time, immense progress has been made in the efficiency of the algorithms used. Take the case of the human shape recognition for mining operations. Distinguishing between a human and a car is easy. The computer can recognize that humans have distinct shapes (such as legs or arms) and that vehicles do not. Distinguishing a human from another mammal is much more difficult (although nonhuman mammals are not common occurrences in mines). The same goes for telling the difference between a pickup truck and a van. You can tell when you see one, but training a machine to differentiate them requires more than basic shape recognition.

This is where neural networks come into the picture. Neural networks are ML methods that mimic the way the human brain works. When you look at a human figure, multiple zones of your brain are activated to recognize colors, movements, facial expressions, and so on. Your brain combines these elements to conclude that the shape you are seeing is human. Neural networks mimic the same logic. The information goes through different algorithms (called *units*), each of which is in charge of processing an aspect of the information. The resulting value of one unit computation can be used directly or fed into another unit for further processing to occur. In this case, the neural network is said to have several layers. For example, a neural network processing human image recognition may have two units in a first layer that determines whether the image has straight lines and sharp angles—because vehicles commonly have straight lines and sharp angles, and human figures do not. If the image passes the first layer successfully (because there are no or only a small percentage of sharp angles and straight lines), a second layer may look for different features (presence of face, arms, and so on), and then a third layer might compare the image to images of various animals and conclude that the shape is a human (or not). The great efficiency of neural networks is that each unit processes a simple test, and therefore computation is quite fast. This model is demonstrated in Figure 7-6.

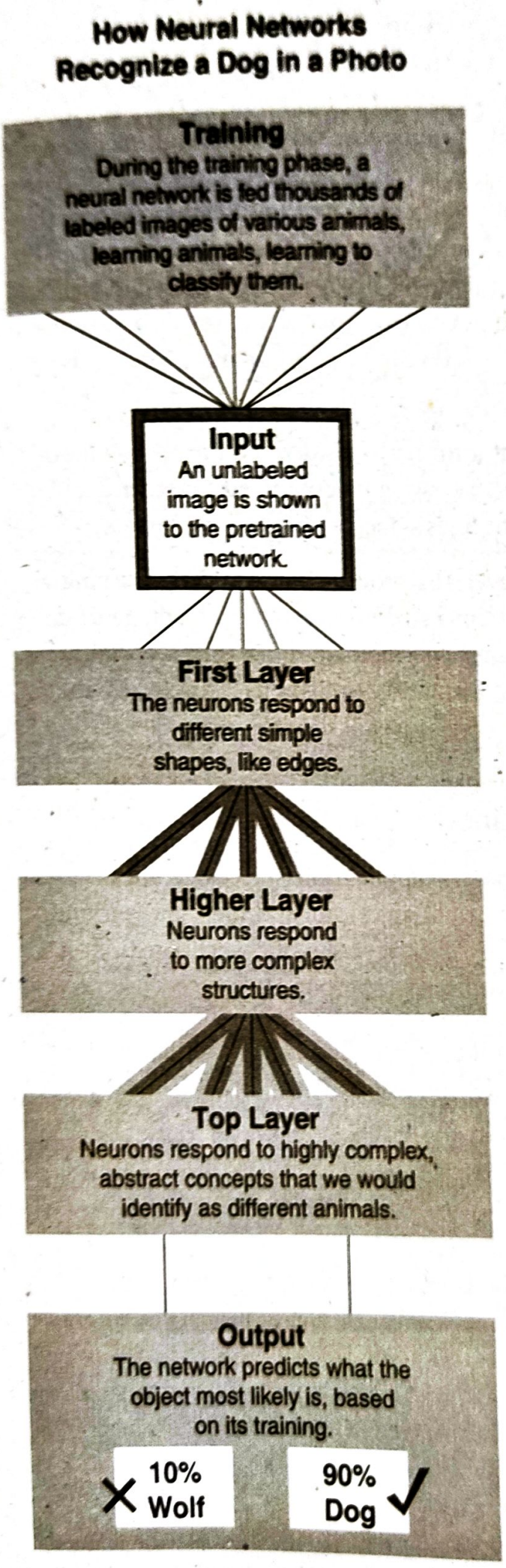

Figure 7-6 *Neural Network Example*

By contrast, old supervised ML techniques would compare the human figure to potentially hundreds of thousands of images during the training phase, pixel by pixel, making them difficult and expensive to implement (with a lot of training needed) and slow to operate. Neural networks have been the subject of much research work. Multiple research and optimization efforts have examined the number of units and layers, the type of data processed at each layer, and the type and combination of algorithms used to process the data to make processing more efficient for specific applications. Image processing can be optimized with certain types of algorithms that may not be optimal for crowd movement classification. Another algorithm may be found in this case that would revolutionize the way these movements are processed and analyzed. Possibilities are as numerous as the applications where they can be used.

In a sense, neural networks rely on the idea that information is divided into key components, and each component is assigned a weight. The weights compared together decide the classification of this information (no straight lines + face + smile = human).

When the result of a layer is fed into another layer, the process is called deep learning ("deep" because the learning process has more than a single layer). One advantage of deep learning is that having more layers allows for richer intermediate processing and representation of the data. At each layer, the data can be formatted to be better utilized by the next layer. This process increases the efficiency of the overall result.

6. Explain big data analytics tools & techniques?

Big Data Analytics Tools and Technology

It is a common mistake for individuals new to the world of data management to use the terms *big data* and *Hadoop* interchangeably. Though it's true that Hadoop is at the core of many of today's big data implementations, it's not the only piece of the puzzle. Big data analytics can consist of many different software pieces that together collect, store,

manipulate, and analyze all different data types. It helps to better understand the landscape by defining what big data is and what it is not. Generally, the industry looks to the "three *Vs*" to categorize big data:

- **Velocity:** *Velocity* refers to how quickly data is being collected and analyzed. Hadoop Distributed File System is designed to ingest and process data very quickly. Smart objects can generate machine and sensor data at a very fast rate and require database or file systems capable of equally fast ingest functions.
- **Variety:** *Variety* refers to different types of data. Often you see data categorized as structured, semi-structured, or unstructured. Different database technologies may only be capable of accepting one of these types. Hadoop is able to collect and store all three types. This can be beneficial when combining machine data from IoT devices that is very structured in nature with data from other sources, such as social media or multimedia, that is unstructured.
- **Volume:** *Volume* refers to the scale of the data. Typically, this is measured from gigabytes on the very low end to petabytes or even exabytes of data on the other extreme. Generally, big data implementations scale beyond what is available on locally attached storage disks on a single node. It is common to see clusters of servers that consist of dozens, hundreds, or even thousands of nodes for some large deployments.

The characteristics of big data can be defined by the sources and types of data. First is machine data, which is generated by IoT devices and is typically unstructured data. Second is transactional data, which is from sources that produce data from transactions on these systems, and, have high volume and structured. Third is social data sources, which are typically high volume and structured. Fourth is enterprise data, which is data that is lower in volume and very structured. Hence big data consists of data from all these separate sources.

An additional point to consider while reviewing data sources is the amount of data ingested from each source, which determines the data storage layer design. You should also consider the mechanism to get the data from the ingest systems—namely push or pull. The type of data source—database, file, web service, stream—also needs to be considered as it also determines the structure of data.

Data ingest is the layer that connects data sources to storage. It's the layer that preprocesses, validates, extracts, and stores data temporarily for further processing. There are several patterns to consider for data ingest. First is multisource ingestion, which connects multiple data sources to ingest systems. In this pattern, ingest nodes receive streams of data from multiple sources and do processing before passing the data to intermediate nodes and to final store nodes. This pattern is typically implemented in batch systems and (less often, due to the delay of data availability) in real-time systems.

Data collection and analysis are not new concepts in the industries that helped define IoT. Industrial verticals have long depended on the ability to get, collect, and record data from various processes in order to record trends and track performance and quality.

For example, many industrial automation and control systems feed data into two distinct database types, relational databases and historians. Relational databases, such as Oracle and Microsoft SQL, are good for transactional, or process, data. Their benefit is being able to analyze complex data relationships on data that arrives over a period of time. On the other hand, historians are optimized for time-series data from systems and processes. They are built with speed of storage and retrieval of data at their core, recording each data point in a series with the pertinent information about the system being logged. This data may consist of a sensor reading, the quantity of a material, a temperature reading, or flow data.

Relational databases and historians are mature technologies that have been with us for many years, but new technologies and techniques in the data management market have opened up new possibilities for sensor and machine data. These database technologies broadly fit into a few categories that each have strengths and potential drawbacks when used in an IoT context. The three most popular of these categories are massively parallel processing systems, NoSQL, and Hadoop.

Massively Parallel Processing Databases

Enterprises have used relational databases for storing structured, row and column style data types for decades. Relational databases are often grouped into a broad data storage category called data warehouses. Though they are the centerpiece of most data architectures, they are often used for longer-term archiving and data queries that can often take minutes or hours. An example of this would be asking for all the items produced in the past year that had a particular specification. Depending on the number of items in the database and the complexity of the question being asked, the response could be slow to return.

Massively parallel processing (MPP) databases were built on the concept of the relational data warehouses but are designed to be much faster, to be efficient, and to support reduced query times. To accomplish this, MPP databases take advantage of multiple nodes (computers) designed in a scale-out architecture such that both data and processing are distributed across multiple systems.

MPPs are sometimes referred to as *analytic databases* because they are designed to allow for fast query processing and often have built-in analytic functions. As the name implies, these database types process massive data sets in parallel across many processors and nodes. An MPP architecture (see Figure 7-7) typically contains a single master node that is responsible for the coordination of all the data storage and processing across the cluster. It operates in a "shared-nothing" fashion, with each node containing local processing, memory, and storage and operating independently. Data storage is optimized across the nodes in a structured SQL-like format that allows data analysts to work with the data using common SQL tools and applications. The earlier example of a complex SQL query could be distributed and optimized, resulting in a significantly faster response. Because data stored on MPPs must still conform to this relational structure, it may not be the only database type used in an IoT implementation. The sources and types of data may vary, requiring a database that is more flexible than relational databases allow.

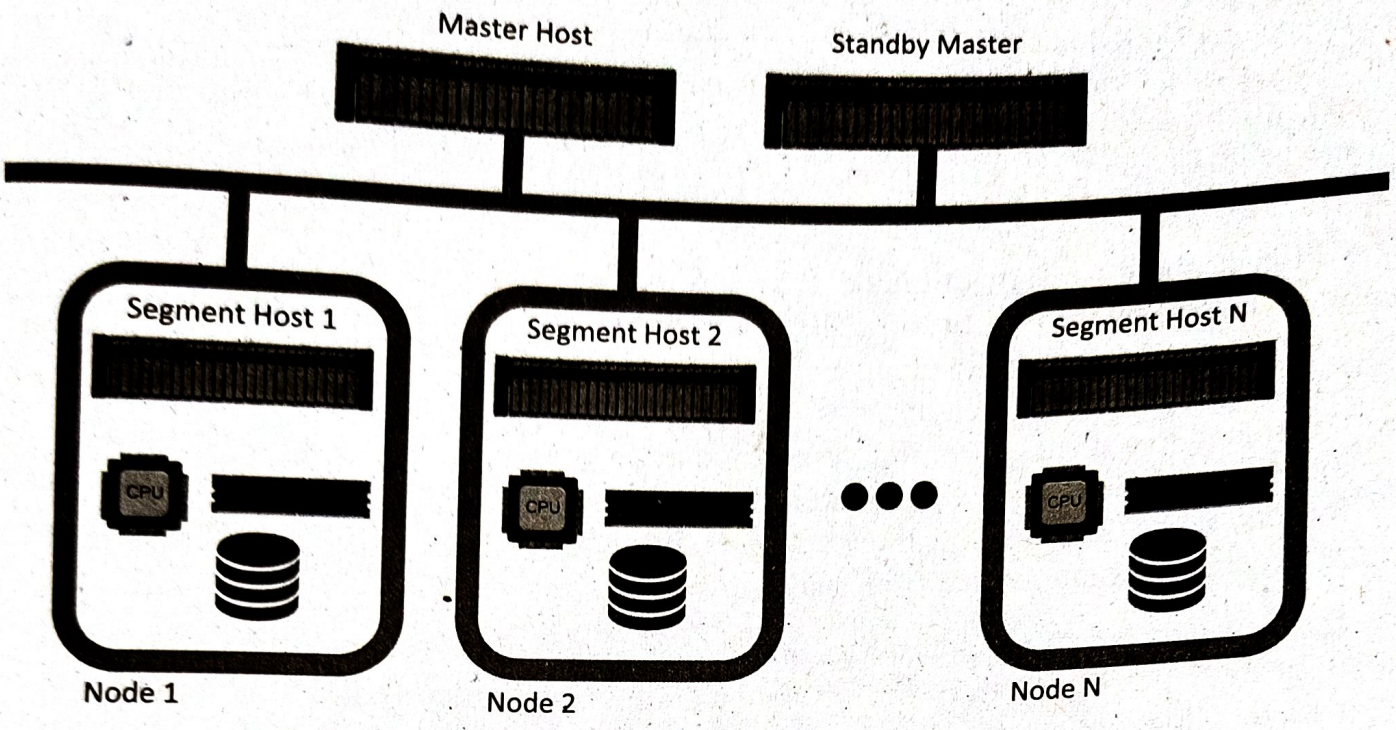

Figure 7-7 *MPP Shared-Nothing Architecture*

NoSQL Databases

NoSQL ("not only SQL") is a class of databases that support semi-structured and unstructured data, in addition to the structured data handled by data warehouses and MPPs. NoSQL is not a specific database technology; rather, it is an umbrella term that encompasses several different types of databases, including the following:

- **Document stores:** This type of database stores semi-structured data, such as XML or JSON. Document stores generally have query engines and indexing features that allow for many optimized queries.
- **Key-value stores:** This type of database stores associative arrays where a key is paired with an associated value. These databases are easy to build and easy to scale.
- **Wide-column stores:** This type of database stores similar to a key-value store, but the formatting of the values can vary from row to row, even in the same table.
- **Graph stores:** This type of database is organized based on the relationships between elements. Graph stores are commonly used for social media or natural language processing, where the connections between data are very relevant.

NoSQL was developed to support the high-velocity, urgent data requirements of modern web applications that typically do not require much repeated use. The original intent was to quickly ingest rapidly changing server logs and clickstream data generated by web-scale applications that did not neatly fit into the rows and columns required by relational databases. Similar to other data stores, like MPPs and Hadoop (discussed later), NoSQL is built to scale horizontally, allowing the database to span multiple hosts, and can even be distributed geographically.

Expanding NoSQL databases to other nodes is similar to expansion in other distributed data systems, where additional hosts are managed by a master node or process. This expansion can be automated by some NoSQL implementations or can be provisioned manually. This level of flexibility makes NoSQL a good candidate for holding machine and sensor data associated with smart objects.

Of the database types that fit under the NoSQL category, key-value stores and document stores tend to be the best fit for what is considered "IoT data." Key-value store is the technology that provides the foundation for many of today's RDBMSs, such as MS SQL, Oracle, and DB2.[3] However, unlike traditional RDBMSs, key-value stores on NoSQL are not limited to a single monolithic system. NoSQL key-value stores are capable of handling indexing and persistence simultaneously at a high rate. This makes it a great choice for time-series data sets, which record a value at a given interval of time, such as a temperature or pressure reading from a sensor.

By allowing the database schema to change quickly, NoSQL document databases tend to be more flexible than key-value store databases. Semi-structured or unstructured data that does not neatly fit into rows and columns can share the same database with organized time-series data. Unstructured data can take many forms; two examples are a photograph of a finished good on a manufacturing line used for QA and a maintenance report from a piece of equipment.

Many NoSQL databases provide additional capabilities, such as being able to query and analyze data within the database itself, eliminating the need to move and process it elsewhere. They also provide a variety of ways to query the database through an API, making it easy to integrate them with other data management applications.

Hadoop

Hadoop is the most recent entrant into the data management market, but it is arguably the most popular choice as a data repository and processing engine. Hadoop was originally developed as a result of projects at Google and Yahoo!, and the original intent for Hadoop was to index millions of websites and quickly return search results for open source search engines. Initially, the project had two key elements:

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

Both of these elements are still present in current Hadoop distributions and provide the foundation for other projects that are discussed later in this chapter.

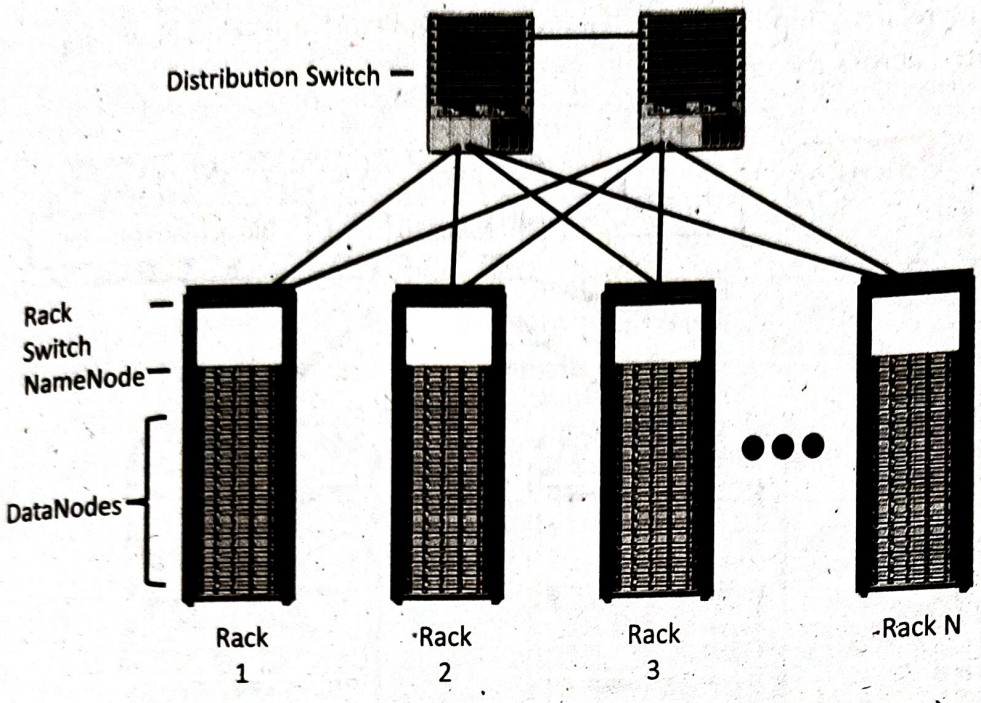

Figure 7-8 *Distributed Hadoop Cluster*

Much like the MPP and NoSQL systems discussed earlier, Hadoop relies on a scale-out architecture that leverages local processing, memory, and storage to distribute tasks and provide a scalable storage system for data. Both MapReduce and HDFS take advantage of this distributed architecture to store and process massive amounts of data and are thus able to leverage resources from all nodes in the cluster. For HDFS, this capability is handled by specialized nodes in the cluster, including NameNodes and DataNodes (see Figure 7-8):

- **NameNodes:** These are a critical piece in data adds, moves, deletes, and reads on HDFS. They coordinate where the data is stored, and maintain a map of where each block of data is stored and where it is replicated. All interaction with HDFS is coordinated through the primary (active) NameNode, with a secondary (standby) NameNode notified of the changes in the event of a failure of the primary. The NameNode takes write requests from clients and distributes those files across the available nodes in configurable block sizes, usually 64 MB or 128 MB blocks. The NameNode is also responsible for instructing the DataNodes where replication should occur.

- **DataNodes:** These are the servers where the data is stored at the direction of the NameNode. It is common to have many DataNodes in a Hadoop cluster to store the data. Data blocks are distributed across several nodes and often are replicated three, four, or more times across nodes for redundancy. Once data is written to one of the DataNodes, the DataNode selects two (or more) additional nodes, based on replication policies, to ensure data redundancy across the cluster. Disk redundancy techniques such as Redundant Array of Independent Disks (RAID) are generally not used for HDFS because the NameNodes and DataNodes coordinate block-level redundancy with this replication technique.

Figure 7-9 shows the relationship between NameNodes and DataNodes and how data blocks are distributed across the cluster.

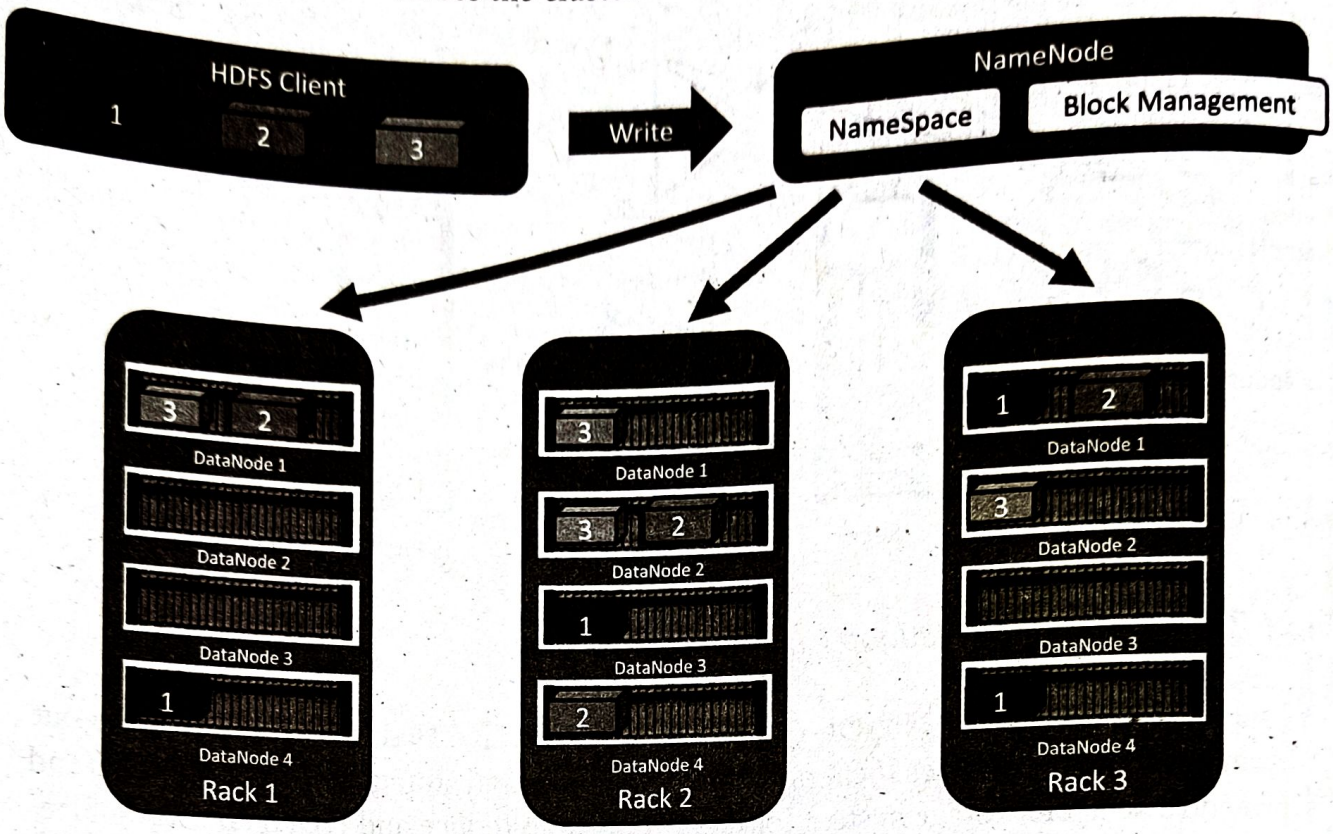

Figure 7-9 *Writing a File to HDFS*

MapReduce leverages a similar model to batch process the data stored on the cluster nodes. Batch processing is the process of running a scheduled or ad hoc query across historical data stored in the HDFS. A query is broken down into smaller tasks and distributed across all the nodes running MapReduce in a cluster. While this is useful for understanding patterns and trending in historical sensor or machine data, it has one significant drawback: time. Depending on how much data is being queried and the complexity of the query, the result could take seconds or minutes to return. If you have a real-time process running where you need a result at a moment's notice, MapReduce is not the right data processing engine for that. (Real-time streaming analytics is discussed later in this chapter.)

YARN

Introduced with version 2.0 of Hadoop, YARN (Yet Another Resource Negotiator) was designed to enhance the functionality of MapReduce. With the initial release, MapReduce was responsible for batch data processing and job tracking and resource management across the cluster. YARN was developed to take over the resource negotiation and job/task tracking, allowing MapReduce to be responsible only for data processing.

With the development of a dedicated cluster resource scheduler, Hadoop was able to add additional data processing modules to its core feature set, including interactive SQL and real-time processing, in addition to batch processing using MapReduce.

7. List & explain key advantages of internet protocol?

The Key Advantages of Internet Protocol

One of the main differences between traditional information technology (IT) and operational technology (OT) is the lifetime of the underlying technologies and products. (For more information on IT and OT, refer to Chapter 1, "What Is IoT?") An entire industrial workflow generally mandates smooth, incremental steps that evolve, with operations itself being the most time- and mission-critical factor for an organization.

One way to guarantee multi-year lifetimes is to define a layered architecture such as the 30-year-old IP architecture. IP has largely demonstrated its ability to integrate small and large evolutions. At the same time, it is able to maintain its operations for large numbers of devices and users, such as the 3 billion Internet users.

Note Using the Internet Protocol suite does not mean that an IoT infrastructure running IP has to be an open or publicly accessible network. Indeed, many existing mission-critical but private and highly secure networks, such as inter-banking networks, military and defense networks, and public-safety and emergency-response networks, use the IP architecture.

Before evaluating the pros and cons of IP adoption versus adaptation, this section provides a quick review of the key advantages of the IP suite for the Internet of Things:

- **Open and standards-based:** Operational technologies have often been delivered as turnkey features by vendors who may have optimized the communications through closed and proprietary networking solutions. The Internet of Things creates a new paradigm in which devices, applications, and users can leverage a large set of devices and functionalities while guaranteeing interchangeability and interoperability,

security, and management. This calls for implementation, validation, and deployment of open, standards-based solutions. While many standards development organizations (SDOs) are working on Internet of Things definitions, frameworks, applications, and technologies, none are questioning the role of the Internet Engineering Task Force (IETF) as the foundation for specifying and optimizing the network and transport layers. The IETF is an open standards body that focuses on the development of the Internet Protocol suite and related Internet technologies and protocols.

- **Versatile:** A large spectrum of access technologies is available to offer connectivity of "things" in the last mile. Additional protocols and technologies are also used to transport IoT data through backhaul links and in the data center. Even if physical and data link layers such as Ethernet, Wi-Fi, and cellular are widely adopted, the history of data communications demonstrates that no given wired or wireless technology fits all deployment criteria. Furthermore, communication technologies evolve at a pace faster than the expected 10- to 20-year lifetime of OT solutions. So, the layered IP architecture is well equipped to cope with any type of physical and data link layers. This makes IP ideal as a long-term investment because various protocols at these layers can be used in a deployment now and over time, without requiring changes to the whole solution architecture and data flow.

- **Ubiquitous:** All recent operating system releases, from general-purpose computers and servers to lightweight embedded systems (TinyOS, Contiki, and so on), have an integrated dual (IPv4 and IPv6) IP stack that gets enhanced over time. In addition, IoT application protocols in many industrial OT solutions have been updated in recent years to run over IP. While these updates have mostly consisted of IPv4 to this point, recent standardization efforts in several areas are adding IPv6. In fact, IP is the most pervasive protocol when you look at what is supported across the various IoT solutions and industry verticals.

- **Scalable:** As the common protocol of the Internet, IP has been massively deployed and tested for robust scalability. Millions of private and public IP infrastructure nodes have been operational for years, offering strong foundations for those not familiar with IP network management. Of course, adding huge numbers of "things" to private and public infrastructures may require optimizations and design rules specific to the new devices. However, you should realize that this is not very different from the recent evolution of voice and video endpoints integrated over IP. IP has proven before that scalability is one of its strengths.

- **Manageable and highly secure:** Communications infrastructure requires appropriate management and security capabilities for proper operations. One of the benefits that comes from 30 years of operational IP networks is the well-understood network management and security protocols, mechanisms, and toolsets that are widely available. Adopting IP network management also brings an operational business application to OT. Well-known network and security management tools are easily leveraged with an IP network layer. However, you should be aware that despite the secure nature of IP, real challenges exist in this area. Specifically, the industry is challenged in securing constrained nodes, handling legacy OT protocols, and scaling operations.

- **Stable and resilient:** IP has been around for 30 years, and it is clear that IP is a workable solution. IP has a large and well-established knowledge base and, more importantly, it has been used for years in critical infrastructures, such as financial and defense networks. In addition, IP has been deployed for critical services, such as voice and video, which have already transitioned from closed environments to open IP standards. Finally, its stability and resiliency benefit from the large ecosystem of IT professionals who can help design, deploy, and operate IP-based solutions.
- **Consumers' market adoption:** When developing IoT solutions and products targeting the consumer market, vendors know that consumers' access to applications and devices will occur predominantly over broadband and mobile wireless infrastructure. The main consumer devices range from smart phones to tablets and PCs. The common protocol that links IoT in the consumer space to these devices is IP.
- **The innovation factor:** The past two decades have largely established the adoption of IP as a factor for increased innovation. IP is the underlying protocol for applications ranging from file transfer and e-mail to the World Wide Web, e-commerce, social networking, mobility, and more. Even the recent computing evolution from PC to mobile and mainframes to cloud services are perfect demonstrations of the innovative ground enabled by IP. Innovations in IoT can also leverage an IP underpinning.

In summary, the adoption of IP provides a solid foundation for the Internet of Things by allowing secured and manageable bidirectional data communication capabilities between all devices in a network. IP is a standards-based protocol that is ubiquitous, scalable, versatile, and stable. Network services such as naming, time distribution, traffic prioritization, isolation, and so on are well-known and developed techniques that can be leveraged with IP. From cloud, centralized, or distributed architectures, IP data flow can be developed and implemented according to business requirements. However, you may wonder if IP is an end-to-end requirement; this is covered in the next section.

8. Explain RPL / encryption & authentication on constraint nodes ?

Authentication and Encryption on Constrained Nodes

IoT security is a complex topic that often spawns discussions and debates across the industry. While IoT security is the focus of Chapter 8, "Securing IoT," we have discussed constrained nodes and networks extensively in this chapter. So it is worth mentioning here the IETF working groups that are focused on their security: ACE and DICE.

ACE

Much like the RoLL working group, the Authentication and Authorization for Constrained Environments (ACE) working group is tasked with evaluating the applicability of existing authentication and authorization protocols and documenting their suitability for certain constrained-environment use cases. Once the candidate solutions are validated, the ACE working group will focus its work on CoAP with the Datagram Transport Layer Security (DTLS) protocol. (The CoAP protocol is covered in Chapter 6, and RFC 6437 defines the DTLS security protocol.) The ACE working group may investigate other security protocols later, with a particular focus on adapting whatever solution is chosen to HTTP and TLS.

The ACE working group expects to produce a standardized solution for authentication and authorization that enables authorized access (Get, Put, Post, Delete) to resources identified by a URI and hosted on a resource server in constrained environments. An unconstrained authorization server performs mediation of the access. Aligned with the initial focus, access to resources at a resource server by a client device occurs using CoAP and is protected by DTLS.

DICE

New generations of constrained nodes implementing an IP stack over constrained access networks are expected to run an optimized IP protocol stack. For example, when implementing UDP at the transport layer, the IETF Constrained Application Protocol (CoAP) should be used at the application layer. (See Chapter 6 for more details on CoAP.)

In constrained environments secured by DTLS, CoAP can be used to control resources on a device. (Constrained environments are network situations where constrained nodes and/or constrained networks are present. Constrained networks and constrained nodes are discussed earlier in this chapter, in the sections "Constrained Nodes" and "Constrained Networks.")

The DTLS in Constrained Environments (DICE) working group focuses on implementing the DTLS transport layer security protocol in these environments. The first task of the DICE working group is to define an optimized DTLS profile for constrained nodes. In addition, the DICE working group is considering the applicability of the DTLS record layer to secure multicast messages and investigating how the DTLS handshake in constrained environments can get optimized.