MODULE -4
Data and Analytics for IoT

4.1 An Introduction to Data Analytics for IoT

In the world of IoT, the creation of massive amounts of data from sensors is common and one of the biggest challenges—not only from a transport perspective but also from a data management standpoint. A great example of the deluge of data that can be generated by IoT is found in the commercial aviation industry and the sensors that are deployed throughout an aircraft.

Modern jet engines are fitted with thousands of sensors that generate a whopping 10GB of data per second. For example, modern jet engines, similar to the one shown in Figure 7-1, maybe equipped with around 5000 sensors. Therefore, a twin engine commercial aircraft with these engines operating on average 8 hours a day will generate over 500 TB of data daily, and this is just the data from the engines! Aircraft today have thousands of other sensors connected to the airframe and other systems. In fact, a single wing of a modern jumbo jet is equipped with 10,000 sensors.

Figure 7-1 *Commercial Jet Engine*

The potential for a petabyte (PB) of data per day per commercial airplane is not farfetched—and this is just for *one* airplane. Across the world, there are approximately 100,000 commercial flights per day. The amount of IoT data coming just from the commercial airline business is overwhelming.

4.1.1 Structured Versus Unstructured Data

Structured data and unstructured data are important classifications as they typically require different toolsets from a data analytics perspective. Figure 7-2 provides a high-level comparison of structured data and unstructured data.

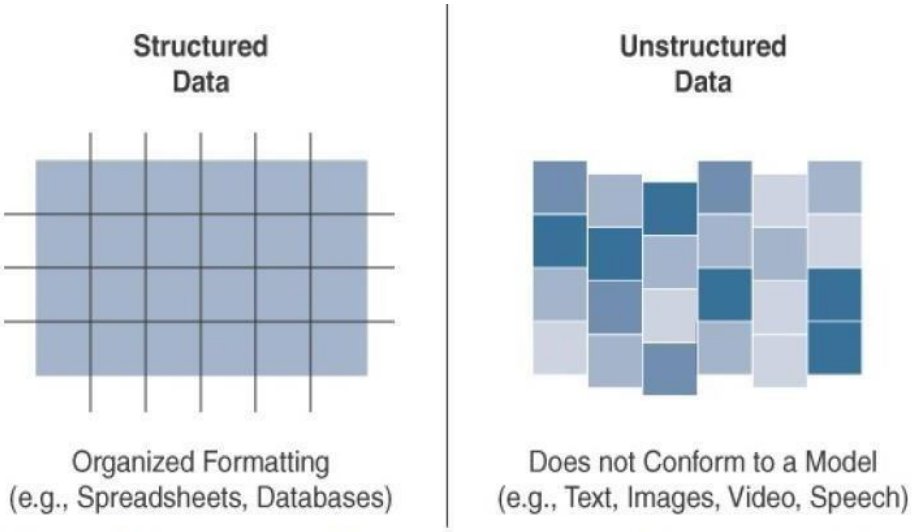

Figure 7-2 *Comparison Between Structured and Unstructured Data*

Structured data means that the data follows a model or schema that defines how the data is represented or organized, meaning it fits well with a traditional relational database management system (RDBMS). In many cases you will find structured data in a simple tabular form—for example, a spreadsheet where data occupies a specific cell and can be explicitly defined and referenced.

Structured data can be found in most computing systems and includes everything from banking transaction and invoices to computer log files and router configurations. IoT sensor data often uses structured values, such as temperature, pressure, humidity, and so on, which are all sent in a known format. Structured data is easily formatted, stored, queried, and processed; for these reasons, it has been the core type of data used for making business decisions.

Unstructured data lacks a logical schema for understanding and decoding the data through traditional programming means. Examples of this data type include text, speech, images, and video. As a general rule, any data that does not fit neatly into a predefined data model is classified as unstructured data.

According to some estimates, around 80% of a business's data is unstructured. Because of this fact, data analytics methods that can be applied to unstructured data, such as cognitive computing and machine learning, are deservedly garnering a lot of attention. With machine learning applications, such as natural language processing (NLP), you can decode speech. With image/facial recognition applications, you can extract critical information from still images and

video.Data in Motion Versus Data at Rest

As in most networks, data in IoT networks is either in transit ("data in motion") or being held or stored ("data at rest"). Examples of data in motion include traditional client/server exchanges, such as web browsing and file transfers, and email. Data saved to a hard drive, storage array, or USB drive is data at rest.

From an IoT perspective, the data from smart objects is considered data in motion as it passes through the network en route to its final destination. This is often processed at the edge, using fog computing. When data is processed at the edge, it may be filtered and deleted or forwarded on for further processing and possible storage at a fog node or in the data center. Data does not come to rest at the edge.

Data at rest in IoT networks can be typically found in IoT brokers or in some sort of storage array at the data center. Myriad tools, especially tools for structured data in relational databases, are available from a data analytics perspective. The best known of these tools is Hadoop. Hadoop not only helps with data processing but also data storage.

4.1.2 IoT Data Analytics Overview

The true importance of IoT data from smart objects is realized only when the analysis of the data leads to actionable business intelligence and insights. Data analysis is typically broken down by the types of results that are produced. As shown in Figure 7-3, there are four types of data analysis results:

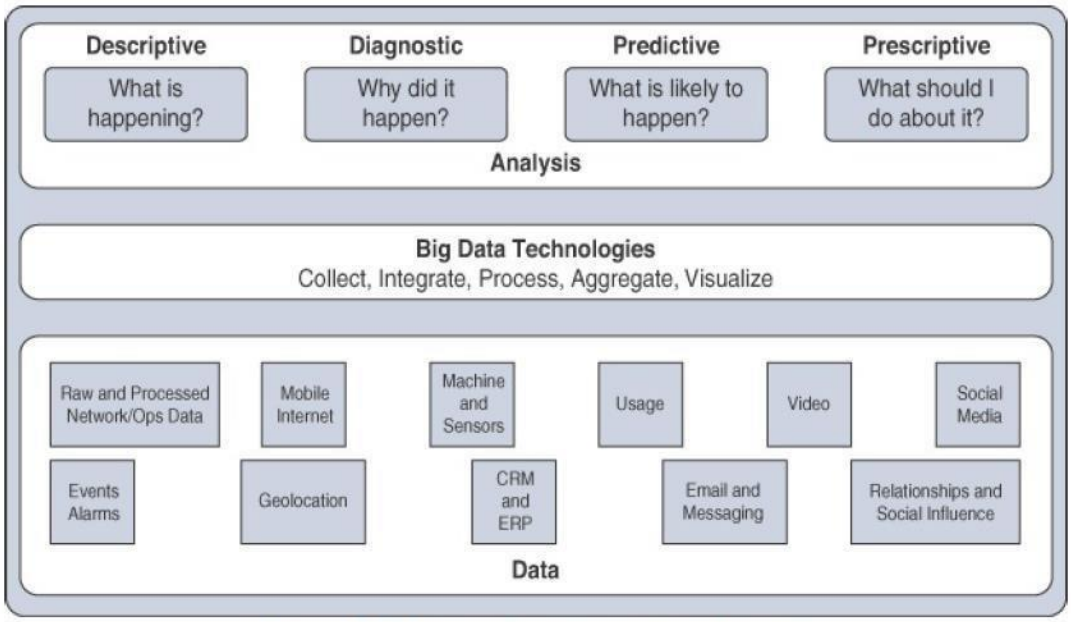

Figure 7-3 *Types of Data Analysis Results*

4.1.2.1 **Descriptive:** Descriptive data analysis tells you what is happening, either now or in the past. For example, a thermometer in a truck engine reports temperature values every second. From a descriptive analysis perspective, you can pull this data at any moment to gain insight into the current operating condition of the truck engine. If the temperature value is too high, then there may be a cooling problem or the engine may be experiencing too much load.

4.1.2.2 **Diagnostic:** When you are interested in the "why," diagnostic data analysis can provide the answer. Continuing with the example of the temperature sensor in the truck engine, you might wonder why the truck engine failed. Diagnostic analysis might show that the temperature of the engine was too high, and the engine overheated. Applying diagnostic analysis across the data generated by a wide range of smart objects can provide a clear picture of why a problem or an event occurred.

4.1.2.3 **Predictive:** Predictive analysis aims to foretell problems or issues before they occur. For example, with historical values of temperatures for the truck engine, predictive analysis could provide an estimate on the remaining life of certain components in the engine. These components could then be proactively replaced before failure occurs. Or perhaps if temperature values of the truck engine start to rise slowly over time, this could indicate the need for an oil change or some other sort of engine cooling maintenance.

4.1.2.4 **Prescriptive:** Prescriptive analysis goes a step beyond predictive and

recommends solutions for upcoming problems. A prescriptive analysis of the temperature data from a truck engine might calculate various alternatives to cost-effectively maintain our truck. These calculations could range from the cost necessary for more frequent oil changes and cooling maintenance to installing new cooling equipment on the engine or upgrading to a lease on a model with a more powerful engine. Prescriptive analysis looks at a variety of factors and makes the appropriate recommendation.

Both predictive and prescriptive analyses are more resource intensive and increase complexity, but the value they provide is much greater than the value from descriptive and diagnostic analysis. Figure 7-4 illustrates the four data analysis types and how they rank as complexity and value increase. You can see that descriptive analysis is the least complex and at the same time offers the least value. On the other end, prescriptive analysis provides the most value but is the most complex to implement. Most data analysis in the IoT space relies on descriptive and diagnostic analysis, but a shift toward predictive and prescriptive analysis is understandably occurring for most businesses and organizations.

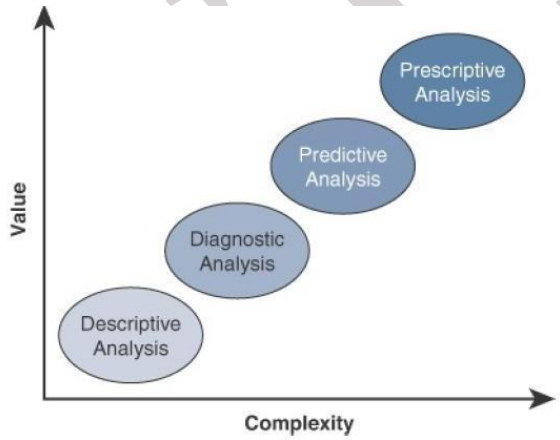

Figure 7-4 *Application of Value and Complexity Factors to the Types of Data Analysis*

As IoT has grown and evolved, it has become clear that traditional data analytics solutions were not always adequate. For example, traditional data analytics typically employs a standard RDBMS and corresponding tools, but the world of IoT is much more demanding. While relational databases are still used for certain data types and applications, they often struggle with the nature of IoT data. IoT data places two specific challenges on a relational database:

4.1.2.5 **Scaling problems:** Due to the large number of smart objects in most IoT networks that continually send data, relational databases can grow incredibly large very quickly. This can result in performance issues that can be costly to

resolve, often requiring more hardware and architecture changes.

4.1.2.6 **Volatility of data:** With relational databases, it is critical that the schema be designed correctly from the beginning. Changing it later can slow or stop the database from operating. Due to the lack of flexibility, revisions to the schema must be kept at a minimum. IoT data, however, is volatile in the sense that the data model is likely to change and evolve over time. A dynamic schema is often required so that data model changes can be made daily or even hourly.

To deal with challenges like scaling and data volatility, a different type of database, known as NoSQL, is being used. Structured Query Language (SQL) is the computer language used to communicate with an RDBMS. As the name implies, a NoSQL database is a database that does not use SQL. It is not set up in the traditional tabular form of a relational database. NoSQL databases do not enforce a strict schema, and they support a complex, evolving data model. These databases are also inherently much more scalable.

4.2 Machine Learning

ML is indeed central to IoT. Data collected by smart objects needs to be analyzed, and intelligent actions need to be taken based on these analyses. Performing this kind of operation manually is almost impossible (or very, very slow and inefficient).

Machines are needed to process information fast and react instantly when thresholds are met. For example, every time a new advance is made in the field of self-driving vehicles, abnormal pattern recognition in a crowd, or any other automated intelligent and machine-assisted decision system, ML is named as the tool that made the advance possible. But ML is not new. It was invented in the middle of the twentieth century and actually fell out of fashion in the 1980s

4.2.1 Machine Learning Overview

ML 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 a vast field but can be simply divided in two main categories: supervised and unsupervised learning.

Supervised Learning

In supervised learning, the machine is trained with input for which there is a known correct answer. For example, suppose that you are training a system to recognize when there is a human in a mine tunnel. A sensor equipped with a basic camera can capture shapes and return them to a computing system that is responsible for determining whether the shape is a human or something else (such as a vehicle, a pile of ore, a rock, a piece of wood, and so on.). With supervised learning techniques,

hundreds or thousands of images are fed into the machine, and each image is labeled (human or nonhuman in this case). This is called the *training set*. An algorithm is used to determine common parameters and common differences between the images. The comparison is usually done at the scale of the entire image, or pixel by pixel. Images are resized to have the same characteristics

(resolution, color depth, position of the central figure, and so on), and each point is analyzed. Human images have certain types of shapes and pixels in certain locations (which correspond to the position of the face, legs, mouth, and so on). Each new image is compared to the set of known "good images," and a deviation is calculated to determine how different the new image is from the average human image and, therefore, the probability that what is shown is a human figure. This process is called *classification*.

After training, the machine should be able to recognize human shapes. Before real field deployments, the machine is usually tested with unlabeled pictures— this is called the validation or the test set, depending on the ML system used—to verify that the recognition level is at acceptable thresholds. If the machine does not reach the level of success expected, more training is needed.

Unsupervised Learning

In some cases, supervised learning is not the best method for a machine to help with a human decision. Suppose that you are processing IoT data from a factory manufacturing small engines. You know that about 0.1% of the produced engines on average need adjustments to prevent later defects, and your task is to identify them before they get mounted into machines and shipped away from the factory. With hundreds of parts, it may be very difficult to detect the potential defects, and it is almost impossible to train a machine to recognize issues that may not be visible. However, you can test each engine and record multiple parameters, such as sound, pressure, temperature of key parts, and so on. Once data is recorded, you can graph these elements in relation to one another (for example, temperature as a function of pressure, sound versus rotating speed over time). You can then input this data into a computer and use mathematical functions to find groups. For example, you may decide to group the engines by the sound they make at a given temperature. A standard function to operate this grouping, *K-means clustering*, finds the mean values for a group of engines (for example, mean value for temperature, mean frequency for sound). Grouping the engines this way can quickly reveal several types of engines that all belong to the same category (for example, small engine of chainsaw type, medium engine of lawnmower type). All engines of the same type produce sounds and temperatures in the same range as the other members of the same group.

There will occasionally be an engine in the group that displays unusual characteristics (slightly out of expected temperature or sound range). This is the engine that you send formanual evaluation. The computing process associated with this determination is called *unsupervised learning*. This type of learning is unsupervised because there is not a "good" or "bad" answer known in advance. It is the variation from a group behavior that allows the computer to learn that something is different. The example of engines is, of course, very simple. In most cases, parameters are multidimensional. In other words, hundreds or thousands of parameters are computed, and

small cumulated deviations in multiple dimensions are used to identify the exception. Figure 7-5 shows an example of such grouping and deviation identification logic. Three parameters are graphed (components 1, 2, and 3), and four distinct groups (clusters) are found. You can see some points that are far from the respective groups. Individual devices that display such "out of cluster" characteristics should be examined more closely individually.

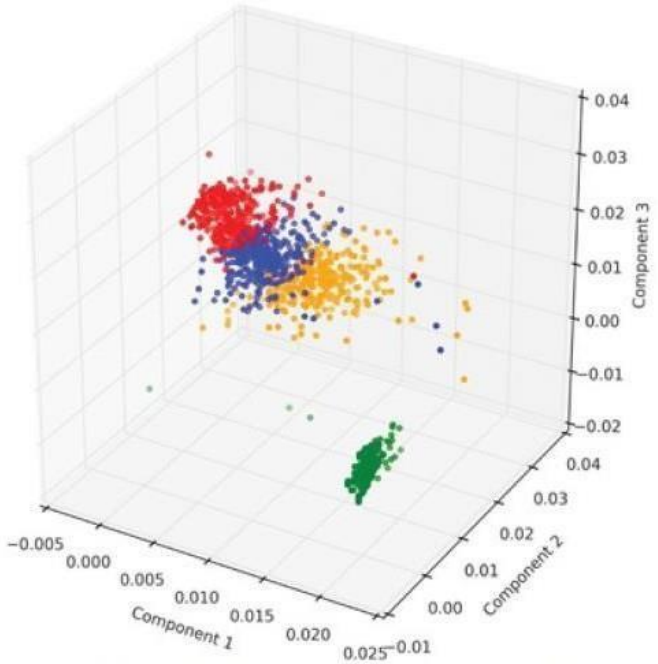

Figure 7-5 *Clustering and Deviation Detection Example*

Neural Networks

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.

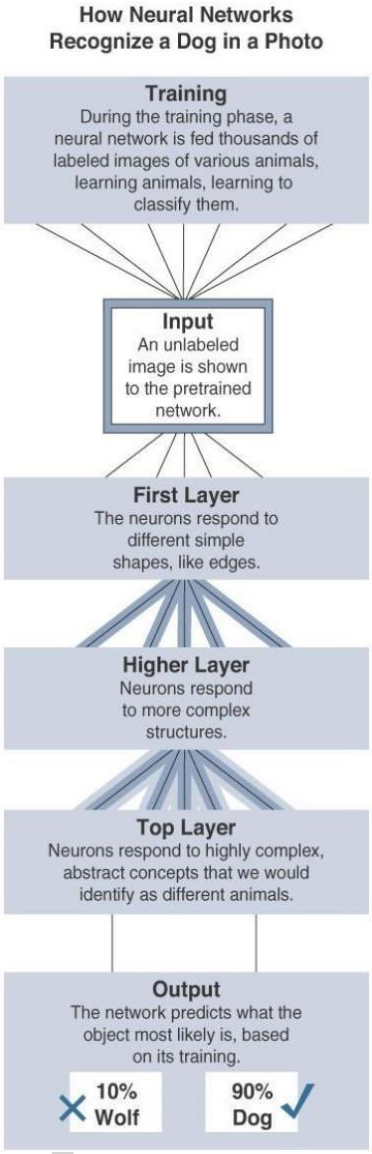

Figure 7-6 *Neural Network Example*

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.

4.2.2 Machine Learning and Getting Intelligence from Big Data

When the principles of machine learning are clear, the application to IoT becomes obvious. The difficulty resides in determining the right algorithm and the right learning model for each use case. Such an analysis goes beyond the scope of this chapter, but it can be useful to organize ML operations into two broad subgroups:

4.2.2.1 **Local learning:** In this group, data is collected and processed locally, either in the sensor itself (the edge node) or in the gateway (the fog node).

4.2.2.2 **Remote learning:** In this group, data is collected and sent to a central computing unit (typically the data center in a specific location or in the cloud), where it is processed.

Regardless of the location where (and, therefore, the scale at which) data is processed, common applications of ML for IoT revolve around four major domains:

4.2.2.3 **Monitoring:** Smart objects monitor the environment where they operate. Data is processed to better understand the conditions of operations. These conditions can refer to external factors, such as air temperature, humidity, or presence of carbon dioxide in a mine, or to operational internal factors, such as the pressure of a pump, the viscosity of oil flowing in a pipe, and so on. ML can be used with monitoring to detect early failure conditions (for example, K-means deviations showing out-of-range behavior) or to better evaluate the environment (such as shape recognition for a robot automatically sorting material or picking goods in a warehouse or a supply chain).

4.2.2.4 **Behavior control:** Monitoring commonly works in conjunction with behavior control. When a given set of parameters reach a target threshold — defined in advance (that is, supervised) or learned dynamically through deviation from mean values (that is, unsupervised)—monitoring functions generate an alarm. This alarm can be relayed to a human, but a more efficient and more advanced system would trigger a corrective action, such as increasing the flow of fresh air in the mine tunnel, turning the robot arm, or reducing the oil pressure in the pipe.

4.2.2.5 **Operations optimization:** Behavior control typically aims at taking corrective actions based on thresholds. However, analyzing data can also lead to changes that improve the overall process. For example, a water purification plant in a smart city can implement a system to monitor the efficiency of the purification process based on which chemical (from company A or company B) is used, at what temperature, and associated to what stirring mechanism (stirring speed and depth). Neural networks can combine multiples of such

units, in one or several layers, to estimate the best chemical and stirring mix for a target air temperature. This intelligence can help the plant reduce its consumption of chemicals while still operating at the same purification efficiency level. As a result of the learning, behavior control results in different machine actions. The objective is not merely to pilot the operations but to improve the efficiency and the result of these operations.

4.2.2.6 **Self-healing, self-optimizing:** A fast-developing aspect of deep learning is the closed loop. ML-based monitoring triggers changes in machine behavior (the change is monitored by humans), and operations optimizations. In turn, the ML engine can be programmed to dynamically monitor and combine new parameters (randomly or semi-randomly) and automatically deduce and implement new optimizations when the results demonstrate a possible gain. The system becomes self-learning and self-optimizing. It also detects new K-means deviations that result in predilection of new potential defects, allowing the system to self-heal. The healing is not literal, as external factors (typically human operators) have to intervene, but the diagnosis is automated. In many cases, the system can also automatically order a piece of equipment that is detected as being close to failure or automatically take corrective actions to avoid the failure (for example, slow down operations, modify a machine's movement to avoid fatigue on a weak link).

4.2.3 Predictive Analytics

Multiple smart objects measure the pull between carriages, the weight on each wheel, and multiple other parameters to offer a form of cruise control optimization for the driver. At the same time, cameras observe the state of the tracks ahead, audio sensors analyze the sound of each wheel on the tracks, and multiple engine parameters are measured and analyzed. All this data can be returned to a data processing center in the cloud that can re-create a virtual twin of each locomotive. Modeling the state of each locomotive and combining this knowledge with anticipated travel and with the states (and detected failures) of all other locomotives of the same type circulating on the tracks of the entire city, province, state, or country allows the analytics platform to make very accurate predictions on what issue is likely to affect each train and each locomotive. Such predictive analysis allows preemptive maintenance and increases the safety and efficiency of operations.

Big Data Analytics Tools and Technology

Generally, the industry looks to the "three *V*s" 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 threetypes. 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 istransactional 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.

Massively Parallel Processing Databases

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 "sharednothing" fashion, with each node containing local processing, memory, and storage and operating independently. Data storage is optimized across the nodes in a structured SQL-likeformat 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.

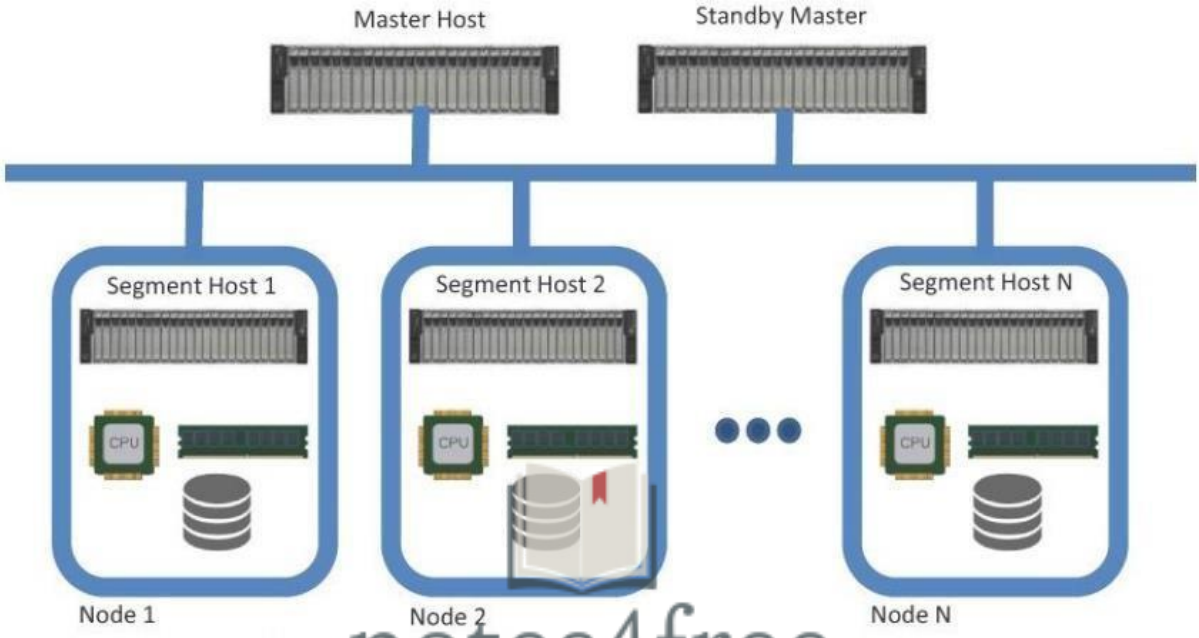

Figure 7-7 *MPP Shared-Nothing Architecture*

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 allowfor 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 otherdata 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.

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.

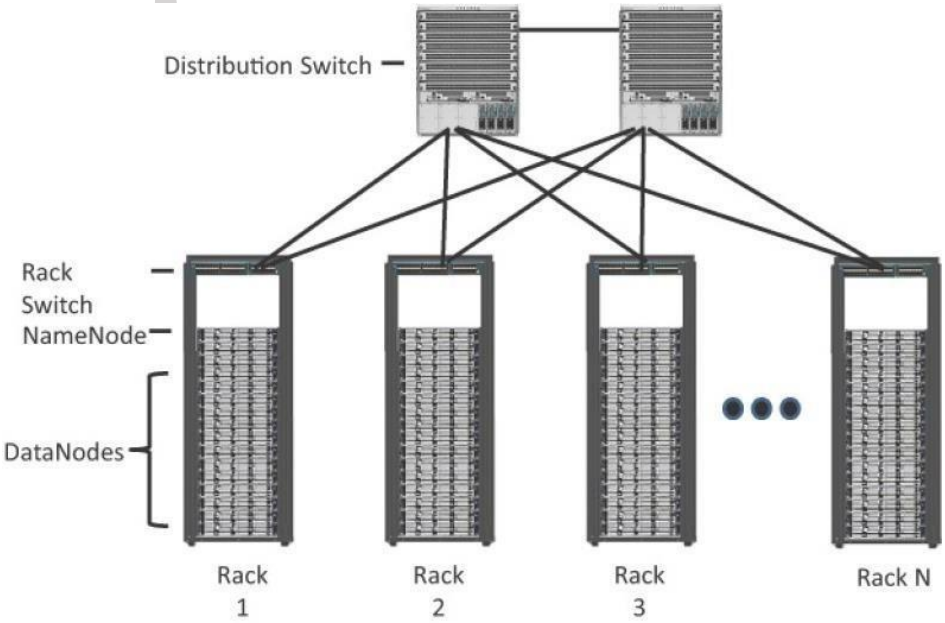

Figure 7-8 *Distributed Hadoop Cluster*

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 NameNodesand DataNodes coordinate blocklevel redundancy with this replication technique. Figure 7-9 shows the relationship between NameNodes and DataNodes and how data blocks are distributed across the cluster.

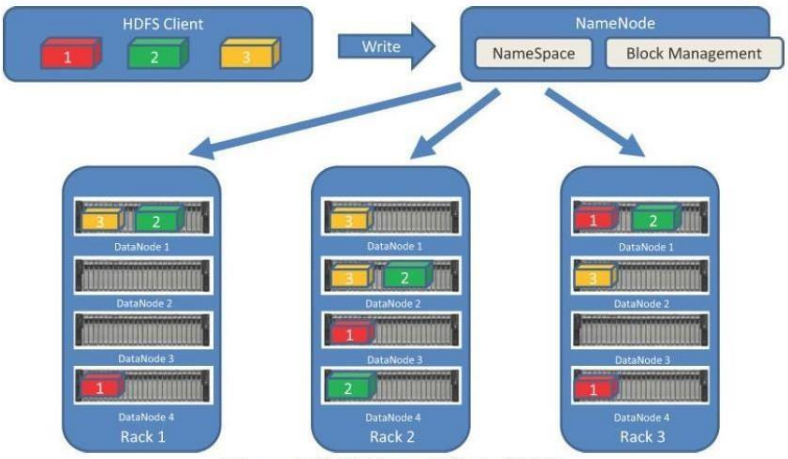

Figure 7-9 *Writing a File to HDFS*

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

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 andreal-time processing, in addition to batch processing using MapReduce.

The Hadoop Ecosystem

As mentioned earlier, Hadoop plays an increasingly big role in the collection, storage, and processing of IoT data due to its highly scalable nature and its ability to work with large volumes of data.
Hadoop now comprises more than 100 software projects under the Hadoop umbrella, capable of nearly every element in the data lifecycle, 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. The following sections describe several of these packages and discuss how they are used to collect or process data.

Apache Kafka

Part of processing real-time events, such as those commonly generated by smart objects, is having them ingested into a processing engine. The process of collecting data from a sensor or log file and preparing it to be processed and analyzed is typically handled by messaging systems. 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. Figure 7-10 shows the data flow from the smart objects

(producers), through a topic in Kafka, to the real-time processing engine. Due to the distributed nature of Kafka, it can run in a clustered configuration that can handle many producers and consumers simultaneously and exchanges 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. The following sections describe several of these packages and discusses how they are used to collect or process data

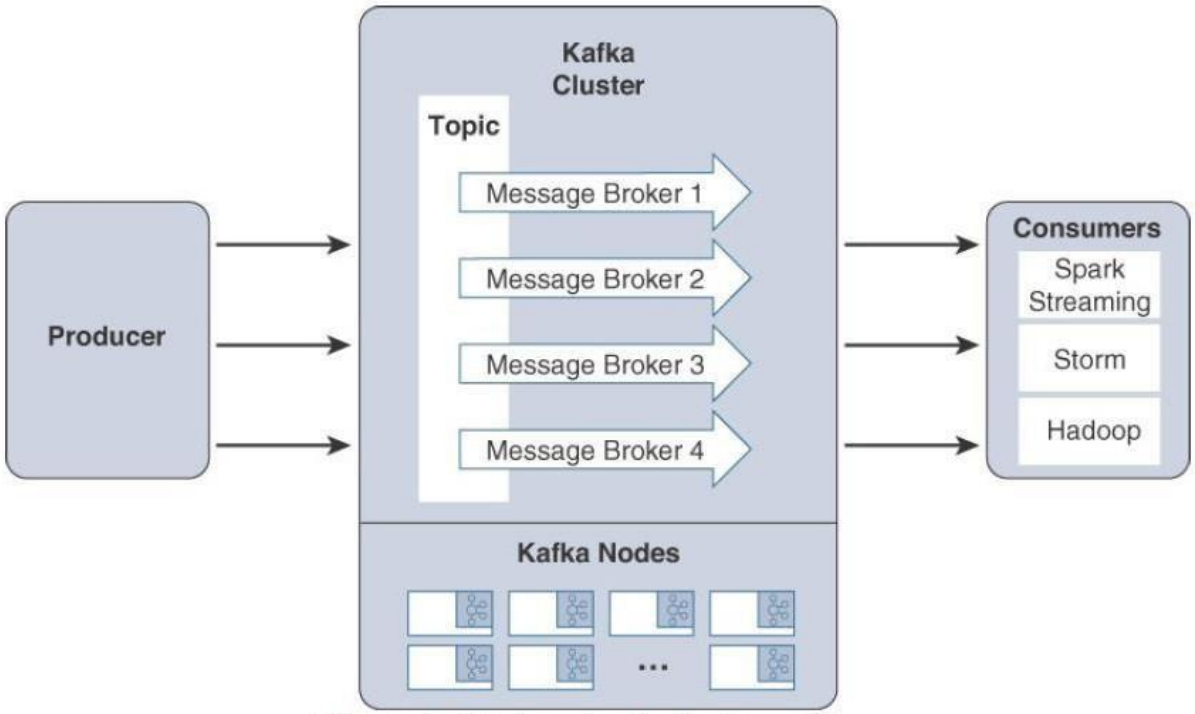

Figure 7-10 *Apache Kafka Data Flow*

Apache Spark

Apache Spark is an in-memory distributed data analytics platform designed to accelerate processes in the Hadoop ecosystem. The "in-memory" characteristic of Spark is what enables it to run jobs very quickly. At each stage of a MapReduce operation, the data is read and written back to the disk, which means latency is introduced through each disk operation. However, with Spark, the processing of this data is moved into high-speed memory, which has significantly lower latency. This speeds the batch processing jobs and also allows for near-real-time processing of events.

Apache Storm and Apache Flink

As you work with the Hadoop ecosystem, you will inevitably notice that different projects are very similar and often have significant overlap with other projects. This is the case with data streaming capabilities. For example, Apache Spark is often used for both distributed streaming analytics and batch processing. Apache Storm and Apache Flink are other Hadoop ecosystem projects designed

for distributed stream processing and are commonly deployed for IoT use cases. Storm can pull data from Kafka and process it in a near-real-time fashion, and so can Apache Flink. This space is rapidly evolving, and projects will continue to gain and lose popularity as they evolve.

Lambda Architecture

Ultimately the key elements of a data infrastructure to support many IoT use cases involves the collection, processing, and storage of data using multiple technologies. Querying both data in motion (streaming) and data at rest (batch processing) requires a combination of the Hadoop ecosystem projects discussed.

One architecture that is currently being leveraged for this functionality is the Lambda Architecture. Lambda is a data management system that consists of two layers for ingesting data (Batch and Stream) and one layer for providing the combined data (Serving). These layers allow for the packages discussed previously, like Spark and MapReduce, to operate on the data independently, focusing on the key attributes for which they are designed and optimized. Data is taken from a message broker, commonly Kafka, and processed by each layer in parallel, and the resulting data is delivered to a data store where additional processing or queries can be run. Figure 7-11 shows this parallel data flow through the Lambda Architecture.

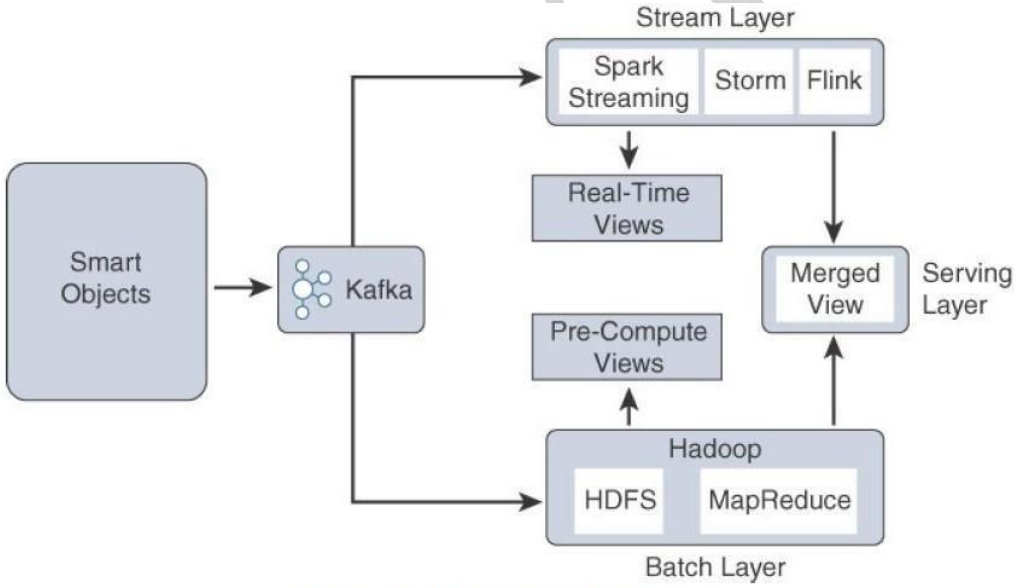

Figure 7-11 *Lambda Architecture*

The Lambda Architecture is not limited to the packages in the Hadoop ecosystem, but due to its breadth and flexibility, many of the packages in the ecosystem fill the requirements of each layer nicely:

- **Stream layer:** This layer is responsible for near-real-time processing of events. Technologies such as Spark Streaming, Storm, or Flink are used to quickly ingest, process, and analyze data on this layer. Alerting and automated actions can be triggered on events that require rapid response or could result in catastrophic outcomes if not handled immediately.
- **Batch layer:** The Batch layer consists of a batch-processing engine and data store. If an organization is using other parts of the Hadoop ecosystem for the other layers, MapReduce and HDFS can easily fit the bill. Other database technologies, such as MPPs, NoSQL, or data warehouses, can also provide what is needed by this layer.
- **Serving layer:** The Serving layer is a data store and mediator that decides which of the ingest layers to query based on the expected result or view into the data. If an aggregate or historical view is requested, it may invoke the Batch layer. If real-time analytics is needed, it may invoke the Stream layer. The Serving layer is often used by the data consumers to access both layers simultaneously.

4.3 Edge Streaming Analytics

One industry where data analytics is used extensively is the world of automobile racing. For example, in Formula One racing, each car has between 150 to 200 sensors that, combined, generate more than 1000 data points per second, resulting in hundreds of gigabytes of raw data per race. The sensor data is transmitted from the car and picked up by track-side wireless sensors. During a race, weather conditions may vary, tire conditions change, and accidents or other racing incidents almost always require an adaptable and flexible racing strategy. As the race develops, decisions such as when to pit, what tires to use, when to pass, and when to slow down all need to be made in seconds. Teams have found that enormous insights leading to better race results can be gained by analyzing data on the fly—and the data may come from many different sources, including trackside sensors, car telemetry, and weather reports.

Comparing Big Data and Edge Analytics

From a business perspective, streaming analytics involves acting on data that is generated while it is still valuable, before it becomes stale. For example, roadway sensors combined with GPS way finding apps may tell a driver to avoid a certain highway due to traffic. This data is valuable for only a small window of time. Historically, it may be interesting to see how many traffic accidents or blockages have occurred on a certain segment of highway or to predict congestion based on past traffic data. However, for the driver in traffic receiving this information, if the data is not acted upon immediately, the data has little value.

From a security perspective, having instantaneous access to analyzed and preprocessed data at the edge also allows an organization to realize anomalies in its network so those anomalies can be quickly contained before spreading to the rest of the network.

To summarize, the key values of edge streaming analytics include the following:

- **Reducing data at the edge:** The aggregate data generated by IoT devices is generally in proportion to the number of devices. The scale of these devices is likely to be huge, and so is the quantity of data they generate. Passing all this data to the cloud is inefficient and is unnecessarily expensive in terms of bandwidth and network infrastructure.

Analysis and response at the edge: Some data is useful only at the edge (such as a factory control feedback system). In cases such as this, the data is best analyzed and acted upon where it is generated.

- **Time sensitivity:** When timely response to data is required, passing data to the cloud for future processing results in unacceptable latency. Edge analytics allows immediate responses to changing conditions.

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.

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 directly at the edge, in the fog, or in the cloud data center. There are no hardand-fast rules dictating where analytics should be done, but there are a few guiding principles. We have already discussed the value of reducing the data at the edge, as well as the value of analyzing information so it can be responded to before it gets stale. There is also value in stepping back from the edge to gain a wider view with more data. It's hard to see the forest when you are standing in the middle of it staring at a tree. In other words, sometimes better insights can be gained and data responded to more intelligently when we step back from the edge and look at a wider data set.

Figure 7-15 shows an example of an oil drilling company that is measuring both pressure and temperature on an oil rig. While there may be some value in doing analytics directly on the edge, in this example, the sensors communicate via MQTT through a message broker to the fog analytics node, allowing a broader data set.

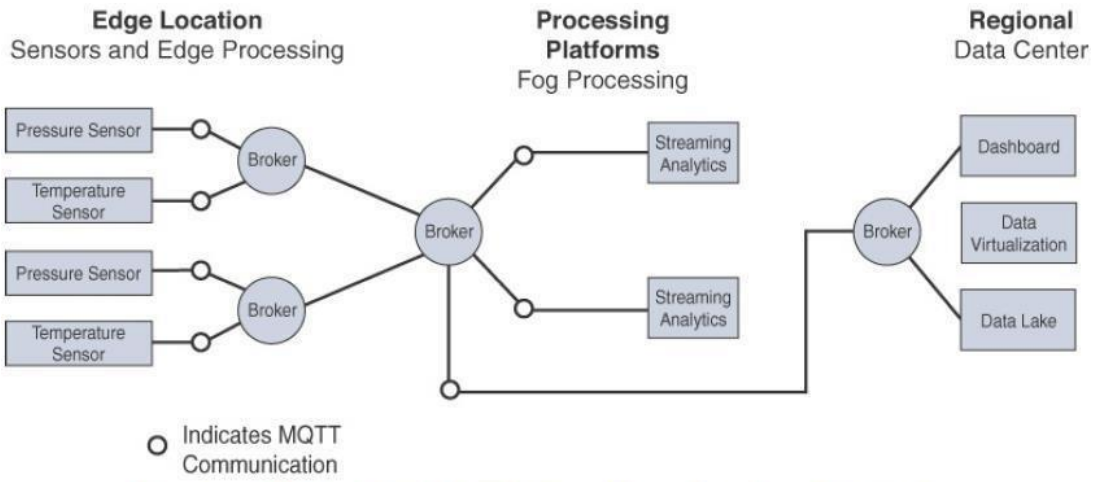

Figure 7-15 *Distributed Analytics Throughout the IoT System*

Network Analytics

Network analytics has the power to analyze details of communications patterns made by protocols and correlate this across the network. It allows you to understand what should be considered normal behavior in a network and to quickly identify anomalies that suggest network problems due to suboptimal paths, intrusive malware, or excessive congestion. Analysis of traffic patterns is one of the most powerful tools in an IoT network engineer's troubleshooting arsenal.

This behavior represents a key aspect that can be leveraged when performing network analytics: 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 (such as an unusual traffic spike for a normally quiet protocol) for both centralized or distributed architectures, such as fog computing.

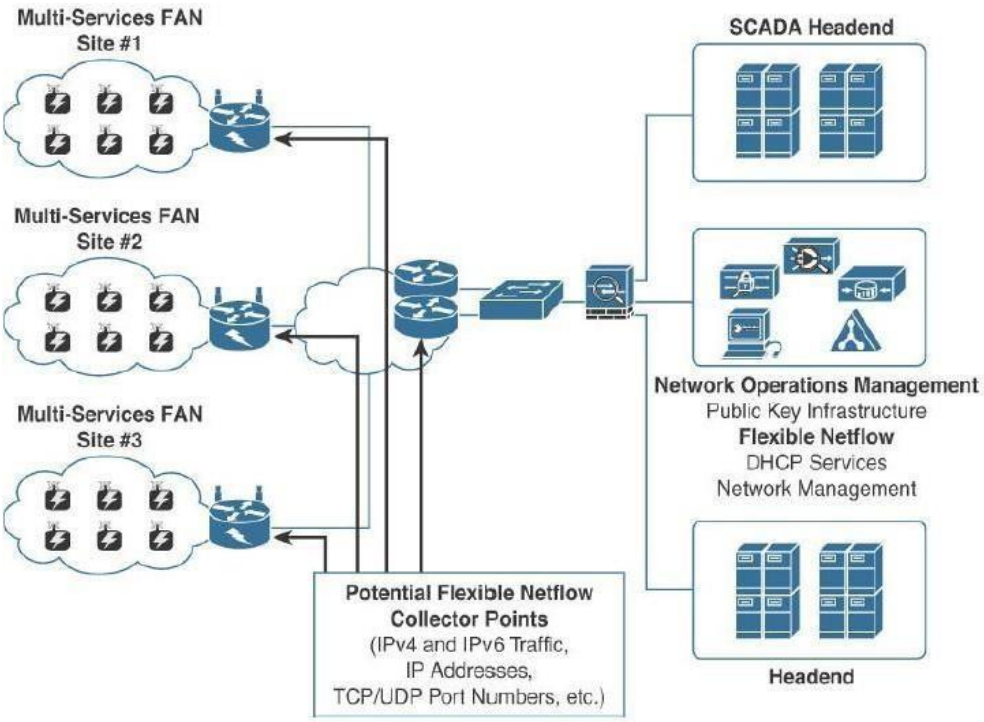

Figure 7-16 *Smart Grid FAN Analytics with NetFlow Example*

Consider that an IoT device sends its traffic to specific servers, either directly to an application or an IoT broker with the data payload encapsulated in a given protocol. This represents a pair of source and destination addresses, as well as application layer–dependent TCP or UDP port numbers, which can be used for network analytics.

4.4 Securing IoT

This chapter provides a historical perspective of OT security, how it has evolved, and some of the common challenges it faces. It also details some of the key differences between securing IT and OT environments. Finally, it explores a number of practical steps for creating a more secure industrial environment, including best practices in introducing modern IT network security into legacy industrial environments.

Common Challenges in OT Security

The security challenges faced in IoT are by no means new and are not limited to specific industrial environments. The following sections discuss some of the common challenges faced in IoT.

Erosion of Network Architecture

There is a wide variety in secured network designs within and across different industries. For

example, power utilities have a strong history of leveraging modern technologies for operational activities, and in North America there are regulatory requirements in place from regulatory authorities, such as North American Electric Reliability Corporation's (NERC's) Critical Infrastructure Protection (CIP)

Pervasive Legacy Systems

Due to the static nature and long lifecycles of equipment in industrial environments, many operational systems may be deemed legacy systems. For example, in a power utility environment, it is not uncommon to have racks of old mechanical equipment still operating alongside modern intelligent electronic devices (IEDs). In many cases, legacy components are not restricted to isolated network segments but have now been consolidated into the IT operational environment. From a security perspective, this is potentially dangerous as many devices may have historical vulnerabilities or weaknesses that have not been patched and updated, or it may be that patches are not even available due to the age of the equipment.

Insecure Operational Protocols

The structure and operation of most of these protocols is often publicly available. While they may have been originated by a private firm, for the sake of interoperability, they are typically published for others to implement. Thus, it becomes a relatively simple matter to compromise the protocols themselves and introduce malicious actors that may use them to compromise control systems for either reconnaissance or attack purposes that could lead to undesirable impacts in normal system operation.

Device Insecurity

Beyond the communications protocols that are used and the installation base of legacy systems, control and communication elements themselves have a history of vulnerabilities.

To understand the nature of the device insecurity, it is important to review the history of what vulnerabilities were discovered and what types of devices were affected. A review of the time period 2000 to 2010 reveals that the bulk of discoveries were at the higher levels of the operational network, including control systems trusted to operate plants, transmission systems, oil pipelines, or whatever critical function is in use.

4.5 How IT and OT Security Practices and Systems Vary

The differences between an enterprise IT environment and an industrial-focused OT deployment are important to understand because they have a direct impact on the security practice applied to them.

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 value, 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 ITcentric solutions being introduced to support operational activities. For example, systems such as firewalls and intrusion prevention systems (IPS) are being used in IoT networks.

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):

4.5.1 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.

4.5.2 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.

4.5.3 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.

4.5.4 Safety zone

- **Safety-critical:** This level includes devices, sensors, and other equipment used to manage the safety functions of the control system.

OT Network Characteristics Impacting Security

While IT and OT networks are beginning to converge, they still maintain many divergent characteristics in terms of how they operate and the traffic they handle. These differences influence how they are treated in the context of a security strategy. For example, compare the nature of how traffic flows across IT and OT networks:

4.5.5 **IT networks:** In an IT environment, there are many diverse data flows. The communication data flows that emanate from a typical IT endpoint travel relatively far. They frequently traverse the network through layers of switches and eventually make their way to a set of local or remote servers, which they may connect to directly.

4.5.6 **OT networks:** By comparison, in an OT environment (Levels 0–3), there are typically two types of operational traffic. The first is local traffic that may be contained within a specific package or area to provide local monitoring and closed-loop control. This is the traffic that is used for realtime (or near-real-time) processes and does not need to leave the process control levels.

Security Priorities: Integrity, Availability, and Confidentiality

In the IT business world, there are legal, regulatory, and commercial obligations to protect data, especially data of individuals who may or may not be employed by the organization. This emphasis on privacy focuses on the confidentiality, integrity, and availability of the data—not necessarily on a system or a physical asset. The impact of losing a compute device is considered minimal compared to the information that it could hold or provide access to. By way of comparison, in the OT world, losing a device due to a security vulnerability means production stops, and the company cannot perform its basic operation. Loss of information stored on these devices is a lower concern, but there are certainly confidential data sets in the operating environment that may have economic impacts, such as formulations and processes.

Security Focus

Security focus is frequently driven by the history of security impacts that an organization has experienced. In an IT environment, the most painful experiences have typically been intrusion campaigns in which critical data is extracted or corrupted. The result has been a significant investment in capital goods and human power to reduce these external threats and minimize potential internal malevolent actors. In the OT space, the history of loss due to external actors has not been as long, even though the potential for harm on a human scale is clearly significantly higher. The result is that the security events that have been experienced have come more from human error than external attacks. Interest and investment in industrial security have primarily been in the standard access control layers. Where OT has diverged, to some degree, is to emphasize the application layer control between the higher-level controller layer and the receiving operating layer. Later in this chapter you will learn more about the value and risks associated with this approach.

4.6 Formal Risk Analysis Structures: OCTAVE and FAIR

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.

OCTAVE

OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) 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

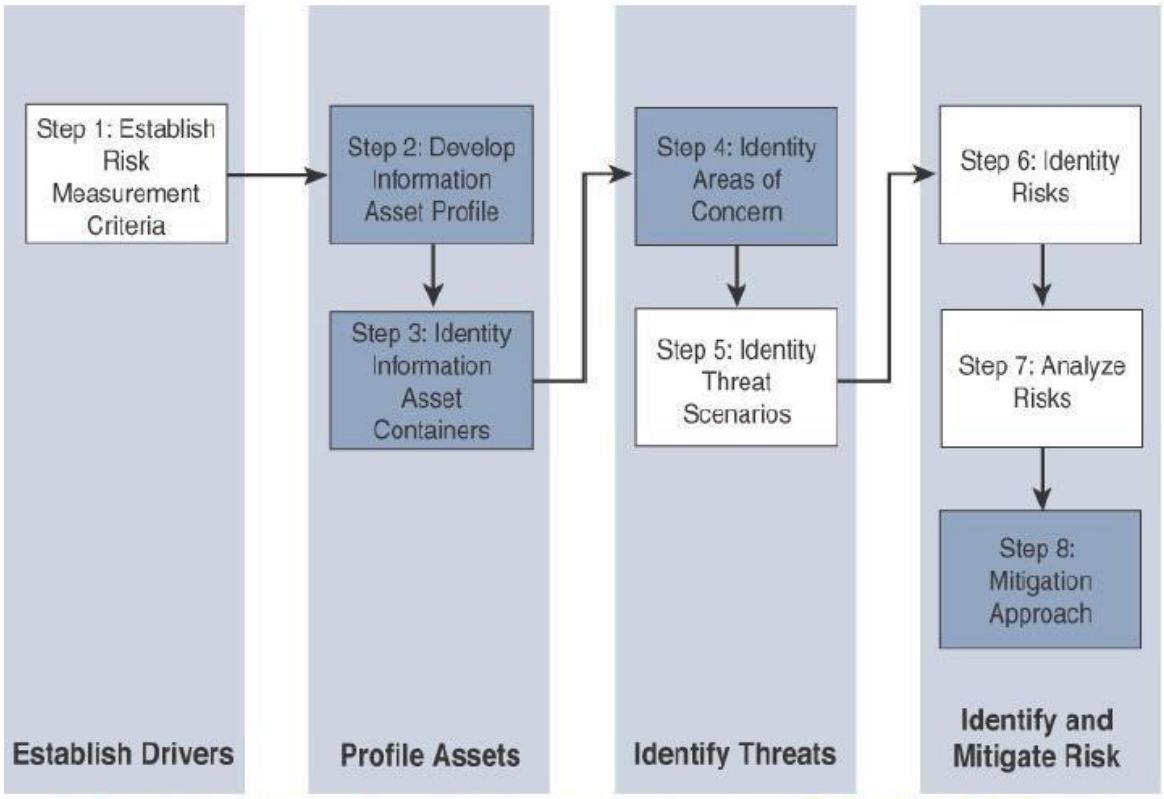

Figure 8-5 *OCTAVE Allegro Steps and Phases (see https://blog.compass-*

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.

FAIR

FAIR (Factor Analysis of Information Risk) is a technical standard for risk definition from The Open Group. While information security is the focus, much as it is for OCTAVE, FAIR has clear applications within operational technology. Like OCTAVE, it also allows for non-malicious actors as a potential cause for harm, but it goes to greater lengths to emphasize the point. For many operational groups, it is a welcome acknowledgement of existing contingency planning. Unlike with OCTAVE, there is a significant emphasis on naming, with risk taxonomy definition as a very specific target.

FAIR places emphasis on both unambiguous definitions and the idea that risk and associated attributes are measurable. Measurable, quantifiable metrics are a key area of emphasis, which should lend itself well to an operational world with a richness of operational data. At its base, FAIR has a definition of risk as the probable frequency and probable magnitude of loss. With

this definition, a clear hierarchy of sub-elements emerges, with one side of the taxonomy focused on frequency and the other on magnitude.

Loss even frequency is the result of a threat agent acting on an asset with a resulting loss to the organization. This happens with a given frequency called the threat event frequency (TEF), in which a specified time window becomes a probability. There are multiple sub-attributes that define frequency of events, all of which can be understood with some form of measurable metric. Threat event frequencies are applied to a vulnerability. *Vulnerability* here is not necessarily some compute asset weakness, but is more broadly defined as the probability that the targeted asset will fail as a result of the actions applied. There are further sub-attributes here as well.

4.7 The Phased Application of Security in an Operational Environment

It is a security practitioner's goal to safely secure the environment for which he or she is responsible. For an operational technologist, this process is different because the priorities and assets to be protected are highly differentiated from the better-known IT environment.

Secured Network Infrastructure and Assets

Given that networks, compute, or operational elements in a typical IoT or industrial system have likely been in place for many years and given that the physical layout largely defines the operational process, this phased approach to introducing modern network security begins with very modest, non-intrusive steps.

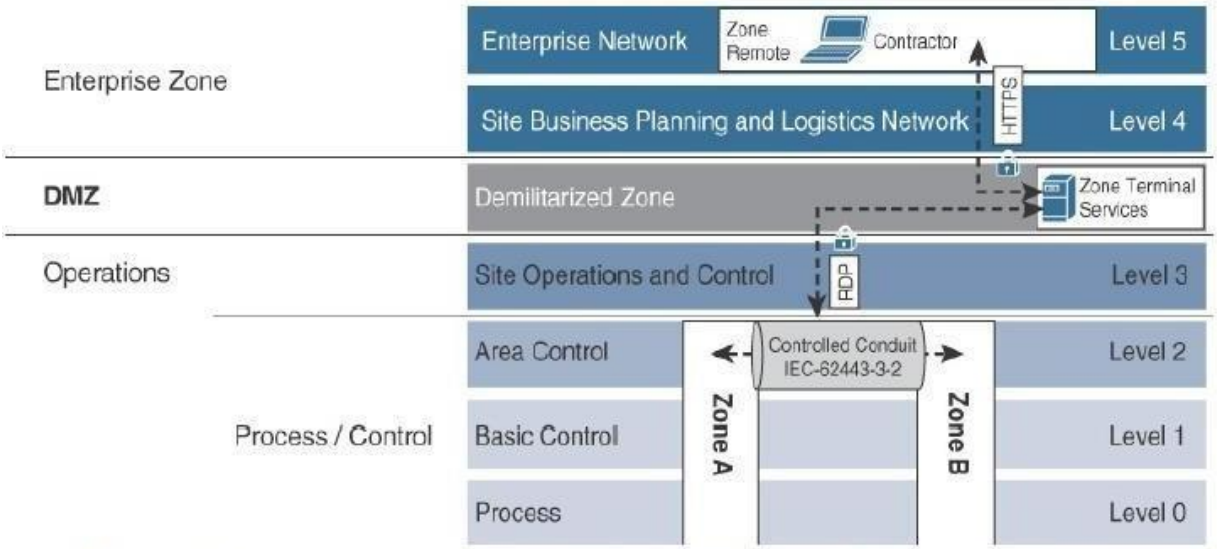

Figure 8-6 *Security Between Levels and Zones in the Process Control Hierarchy Model*

Normal network discovery processes can be highly problematic for older networking equipment. In fact, the discovery process in pursuit of improved safety, security, and operational state can result in degradation of all three.

Deploying Dedicated Security Appliances

The next stage is to expand the security footprint with focused security functionality. The goal is to provide visibility, safety, and security for traffic within the network. Visibility provides an understanding of application and communication behavior. With visibility, you can set policy actions that reflect the desired behaviors for inter-zone and conduit security. While network elements can provide simplified views with connection histories or some kind of flow data, you get a true understanding when you look within the packets on the network. This level of visibility is typically achieved with deep packet inspection (DPI) technologies such as intrusion detection/prevention systems (IDS/IPS). These technologies can be used to detect many kinds of traffic of interest, from simply identifying what applications are speaking, to whether communications are being obfuscated, to whether exploits are targeting vulnerabilities, to passively identifying assets on the network.

With the goal of identifying assets, an IDS/IPS can detect what kind of assets are present on the network. Passive OS identification programs can capture patterns that expose the base operating systems and other applications communicating on the network. The organizationally unique identifier (OUI) in a captured MAC address, which could have come from ARP table exploration, is yet another means of exposure. Coupled with the physical and historical data mentioned before, this is a valuable tool to expand on the asset inventory without having to dangerously or intrusively prod critical systems.

Higher-Order Policy Convergence and Network Monitoring

Another security practice that adds value to a networked industrial space is convergence, which is the adoption and integration of security across operational boundaries. This means coordinating security on both the IT and OT sides of the organization. Convergence of the IT and OT spaces is merging, or at least there is active coordination across formerly distinct IT and OT boundaries. From a security perspective, the value follows the argument that most new networking and compute technologies coming to the operations space were previously found and established in the IT space. It is expected to also be true that the practices and tools associated with those new technologies are likely to be more mature in the IT space.

There are advanced enterprise-wide practices related to access control, threat detection, and many other security mechanisms that could benefit OT security.

As stated earlier, the key is to adjust the approach to fit the target environment. Several areas are more likely to require some kind of coordination across IT and OT environments. Twosuch areas are remote access and threat detection. For remote access, most large industrial organizations backhaul communication through the IT network. Some communications, such as email and web browsing, are obvious communication types that are likely to touch shared IT infrastructure. Often vendors or consultants who require some kind of remote access to OT assets also traverse the IT side of the

network. Given this, it would be of significant value for an OT security practitioner to coordinate access control policies from the remote initiator across the Internet-facing security layers, through the core network, and to a handoff point at the industrial demarcation and deeper, toward the IoT assets.

The use of common access controls and operational conditions eases and protects network assets to a greater degree than having divergent groups creating ad hoc methods. Using location information, participant device security stance, user identity, and access target attributes are all standard functions that modern access policy tools can make use of. Such sophistication is a relatively new practice in industrial environments, and so, if these functions are available, an OT security practitioner would benefit from coordination with his or her IT equivalents.