A NOVEL APPROACH FOR HOTEL MANAGEMENT SYSTEM USING CASSANDRA

Hima S[1], Varalakshmi P[2] and Surekha Mariam Varghese[3]

Department of Computer Science and Engineering, M.A. College of Engineering, Kothamangalam, Kerala, India

ABSTRACT

Apache Cassandra is a distributed storage system for managing very large amounts of structured data. Cassandra provides highly available service with no single point of failure. Cassandra aims to run on top of an infrastructure of hundreds of nodes possibly spread across different data centers with small and large components fail continuously. Cassandra manages the persistent state in the face of the failures which drives the reliability and scalability of the software systems. Cassandra does not support a full relational data model because it resembles a database and shares many design and implementation strategies. In this paper, discuss an implementation of Cassandra as Hotel Management System application. Cassandra system was designed to run on cheap commodity hardware. Cassandra provides high write throughput and read efficiency.

KEYWORDS

Cassandra, Data model.

1. INTRODUCTION

Apache Cassandra is an open source, distributed, highly available, decentralized, elastically scalable, fault-tolerant, consistent, column-oriented database. Cassandra's distribution design is based on Amazon's Dynamo and its data model on Google's Bigtable. Cassandra was introduced at Facebook; it is now used at some of the most popular sites on the Web [1].Apache Cassandra is a type of NoSQL database designed to handle large amounts of data across many servers. This database provides high availability and no single point of failure.Some of the important points of Apache Cassandra: (1) It is scalable, consistent and fault-tolerant, (2) It is key-value as well as column-oriented database,(3) Its data model is based on Google's Bigtable and distribution design is based on Amazon's Dynamo, (4) Introduced at Facebook, it differs sharply from relational database management systems,(5) Cassandra implements a Dynamo-style replication model, also adds a more powerful "column family" data model, and (6) Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Cassandra has become so popular because of its outstanding technical features. Given below are some of the features of Cassandra:

- Elastic scalability: Cassandra allows adding more hardware to accommodate more customers and more data as per requirement.
- Always on architecture: Cassandra is continuously available for critical business applications that cannot afford single point of failure.
- Fast linear-scale performance: Cassandra increases throughput as the number of nodes in the cluster is increased. Therefore it provides a quick response time.

DOI:10.5121/ijfcst.2016.6301

- Flexible data storage: Cassandra handles all possible data formats including: structured, semi-structured, and unstructured. It can dynamically provide changes to data structures according to user need.
- Easy data distribution: Cassandra provides the flexibility to distribute data where user need by replicating data across multiple data centers.
- Transaction support: Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability (ACID).
- Fast writes: Cassandra was designed to run on cheap commodity hardware. It performs fast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.
- The rest of this paper is organized as follows. Section 2 discusses NoSQL database. Section 3 presents the Cassandra Architecture. Section 4 describes the data model of Cassandra. Section 5 describes the implementation details of Hotel Management System. The conclusion is given in Section 6.

2. EXISTING RELATIONAL DATABASE

Relational Databases are also popular like NoSQL database. But it has various drawbacks. Typically address these problems in one or more of the following ways, sometimes in this order:

Throw hardware at the problem by adding more memory, adding faster processors, and upgrading disks. This is known as v*ertical scaling.*

- When the problems arise again, the answer appears to be similar: now that one box is maxed out, you add hardware in the form of additional boxes in a database cluster. Now the problems are data replication and consistency during regular usage and in failover scenarios.
- Now need to update the configuration of the database management system. This might mean optimizing the channels the database uses to write to the underlying filesystem. Then turn off logging or journaling, which frequently is not a desirable (or, depending on situation, legal) option.
- Having put what attention into the database system, turn to the application. Then try to improve indexes. Also optimize the queries. But presumably at this scale weren't wholly ignorant of index and query optimization, and already had them in pretty good shape. So this becomes a painful process of picking through the data access code to find any opportunities for fine tuning. This might include reducing or reorganizing joins, throwing out resource-intensive features such as XML processing within a stored procedure, and so forth. Of course, presumably doing that XML processing for a reason, so if it do somewhere, move the problem to the application layer, hoping to solve it there and crossing fingers that don't break something else in the meantime.
- Employ a caching layer. For larger systems, this might include distributed caches such as memcached, EHCache, Oracle Coherence, or other related products. Now we have a consistency problem between updates in the cache and updates in the database, which is exacerbated over a cluster.
- It is possible to duplicate some of the data to make it look more like the queries that access it. This process, called denormalization, is antithetical to the five normal forms that characterize the relational model, and violate Codd's 12 Commandments for relational data.

Like Cassandra it also supports ACID properties. ACID is an acronym for Atomic, Consistent, Isolated, Durable, which are the gauges we can use to assess that a transaction has executed properly and that it was successful:

Atomic

Atomic means "all or nothing"; that is, when a statement is executed, every update within the transaction must succeed in order to be called and another related update failed. The common example here is with monetary transfers at an ATM: the transfer requires subtracting money from one account and adding it to another account. This operation cannot be subdivided; they must both succeed.

Consistent

Consistent means that data moves from one correct state to another correct state, with no possibility that readers could view different values that don't make sense together. For example, if a transaction attempts to delete a Customer and her Order history, it cannot leave Order rows that reference the deleted customer's primary key; this is an inconsistent state that would cause errors if someone tried to read those Order records.

Isolated

Isolated means that transactions executing concurrently will not become entangled with each other; they each execute in their own space. That is, if two different transactions attempt to modify the same data at the same time, then one of them will have to wait for the other to complete.

Durable

Once a transaction has succeeded, the changes will not be lost. This doesn't imply another transaction won't later modify the same data; it just means that writers can be confident that the changes are available for the next transaction to work with as necessary.

3. NOSQL DATABASE

A NoSQL database (also called as Not Only SQL) is a database that provides a mechanism to store and retrieve data other than the tabular relations used in relational databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data.

The primary objective of a NoSQL database is to have

- simplicity of design,
- horizontal scaling, and
- finer control over availability.

NoSql databases use different data structures compared to relational databases. It makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it must solve.

4. CASSANDRA ARCHITECTURE

The design goal of Cassandra is to handle big data workloads across multiple nodes without any single point of failure. Cassandra has peer-to-peer distributed system, and data is distributed among all the nodes in a cluster [2].

- All the nodes in a cluster play the same role. Each node is independent and at the same time interconnected to other nodes.
- Each node in a cluster can accept read and write requests, regardless of where the data is actually located in the cluster.
- When a node goes down, read/write requests can be served from other nodes in the network.

4.1. Data Replication in Cassandra

In Cassandra, one or more of the nodes in a cluster act as replicas for a given piece of data. If it is detected that some of the nodes responded with an out-of-date value, Cassandra will return the most recent value to the client. After returning the most recent value, Cassandra performs a read repair in the background to update the stale values.

The figure 1 shows a schematic view of how Cassandra uses data replication among the nodes in a cluster to ensure no single point of failure. Cassandra uses the Gossip Protocol to allow the nodes to communicate with each other and detect any faulty nodes in the cluster.

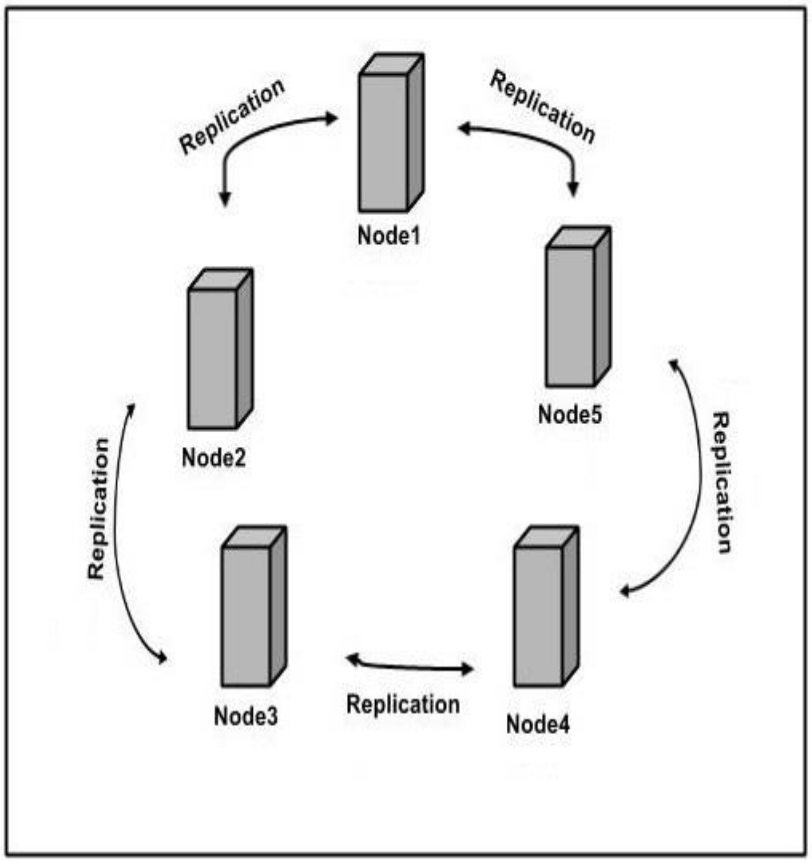

Figure. 1 Schematic view of Cassandra

4.2. Components of Cassandra

The key components of Cassandra are as follows:

- Node: It is the place where data is stored.
- Data center: It is a collection of related nodes.
- Cluster: A cluster is a component that contains one or more data centers.
- Commit log: The commit log is a crash-recovery mechanism in Cassandra. Every write operation is written to the commit log.
- Mem-table: A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem-tables.
- SSTable: It is a disk file to which the data is flushed from the mem-table when its contents reach a threshold value.
- Bloom filter: These are quick, nondeterministic, algorithms for testing whether an element is a member of a set. It is a special kind of cache. Bloom filters are accessed after every query.

4.3. Cassandra Query Language

Users can access Cassandra through its nodes using Cassandra Query Language (CQL). CQL treats the database (Keyspace) as a container of tables. Programmers use cqlsh: a prompt to work with CQL or separate application language drivers.

4.4. Write Operations

Every write activity of nodes is captured by the commit logs written in the nodes. Then the data will be captured and stored in the mem-table. Whenever the mem-table is full, data will be written into the SStable data file. All writes are automatically partitioned and replicated throughout the cluster. Cassandra periodically consolidates the SSTables, deleting unnecessary data.

4.5. Read Operations

During read operations, Cassandra gets values from the mem-table. It checks the bloom filter to find the appropriate SSTable that holds the required data.

5. DATA MODEL

The data model of Cassandra is significantly different from the normal RDBMS [2].

5.1. Cluster

Cassandra database is distributed over several machines that operate together [3]. The outermost container is known as the Cluster. For failure handling, every node contains a replica. In case of a failure, the replica takes charge. Cassandra arranges the nodes in a cluster, in a ring manner, and assigns data to them.

5.2. Keyspace

Keyspace is the outermost container for data in Cassandra. The basic attributes of a Keyspace in Cassandra are:

- Replication factor: It is the number of machines in the cluster that will receive copies of the same data.
- Replica placement strategy: It is the strategy to place replicas in the ring. The different strategies such as simple strategy (rack-aware strategy), old network topology strategy (rack-aware strategy), and network topology strategy (data center-shared strategy) are available.
- Column families: Keyspace is a container for a list of one or more column families. A column family is a container of a collection of rows. Each row contains ordered columns. Column families represent the structure of data. Each keyspace has at least one and often many column families.

6. IMPLEMENTATION DETAILS

The implementation of Apache Cassandra includes installing and configuring Cassandra. Initially download Cassandra from cassandra.apache.org. Copy the folder named cassandra. Move to bin folder. Open the Cassandra.yaml file which is available in the bin folder of the Cassandra folder. Verify that the following configurations.

- data_file_directories **"/var/lib/cassandra/data"**
- commitlog_directory **"/var/lib/cassandra/commitlog"**
- saved_caches_directory **"/var/lib/cassandra/saved_caches"**

Setting the path
Set the path as Cassandra_Home= C:\apache-cassandra-1.2.19
Starting Cassandra
$ cd $CASSANDRA_HOME
$./bin/cassandra –f
Starting cqlsh
Start cqlsh using the command **cqlsh** as shown below. It gives the Cassandra cqlsh prompt as output.
$ cqlsh Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.1.2 | CQL spec 3.2.0 | Native protocol v3]
cqlsh>

An application of Cassandra implementation is Hotel Management System (HMS) [5].Cassandra database is chosen for this application because of its increasing throughput as the number of nodes increases, continuous availability for critical business applications and elastic scalability. Moreover Cassandra handles all possible data formats and distribution of data by replicating data across multiple data centres. Cassandra supports ACID properties and it works on cheap commodity hardware.

In the keyspace of Hotel Management System Figure 2 we have the following column families: Hotel, HotelByCity, Guest, Reservation, PointOfInterest, Room, Room Availability.

In this design, transferred some of the tables, such as Hotel and Guest, to column families. Other tables, such as PointOfInterest, have been denormalized into a super column family. We have created an index in the form of the HotelByCity column family.

We have combined room and amenities into a single column family, Room. The columns such as type and rate will have corresponding values; other columns, such as hot tub, will just use the presence of the column name itself as the value, and be otherwise empty.

Hotel Management System includes details about different hotels, guests who stay in the hotels, availability of rooms for each hotel, and a record of the reservation, which is a certain guest in a certain room for a certain period of time (called the "stay"). Hotels typically also maintain a collection of "points of interest," which are shopping galleries, monuments, museums, parks, or other places near the hotel that guests might like to visit during their stay.

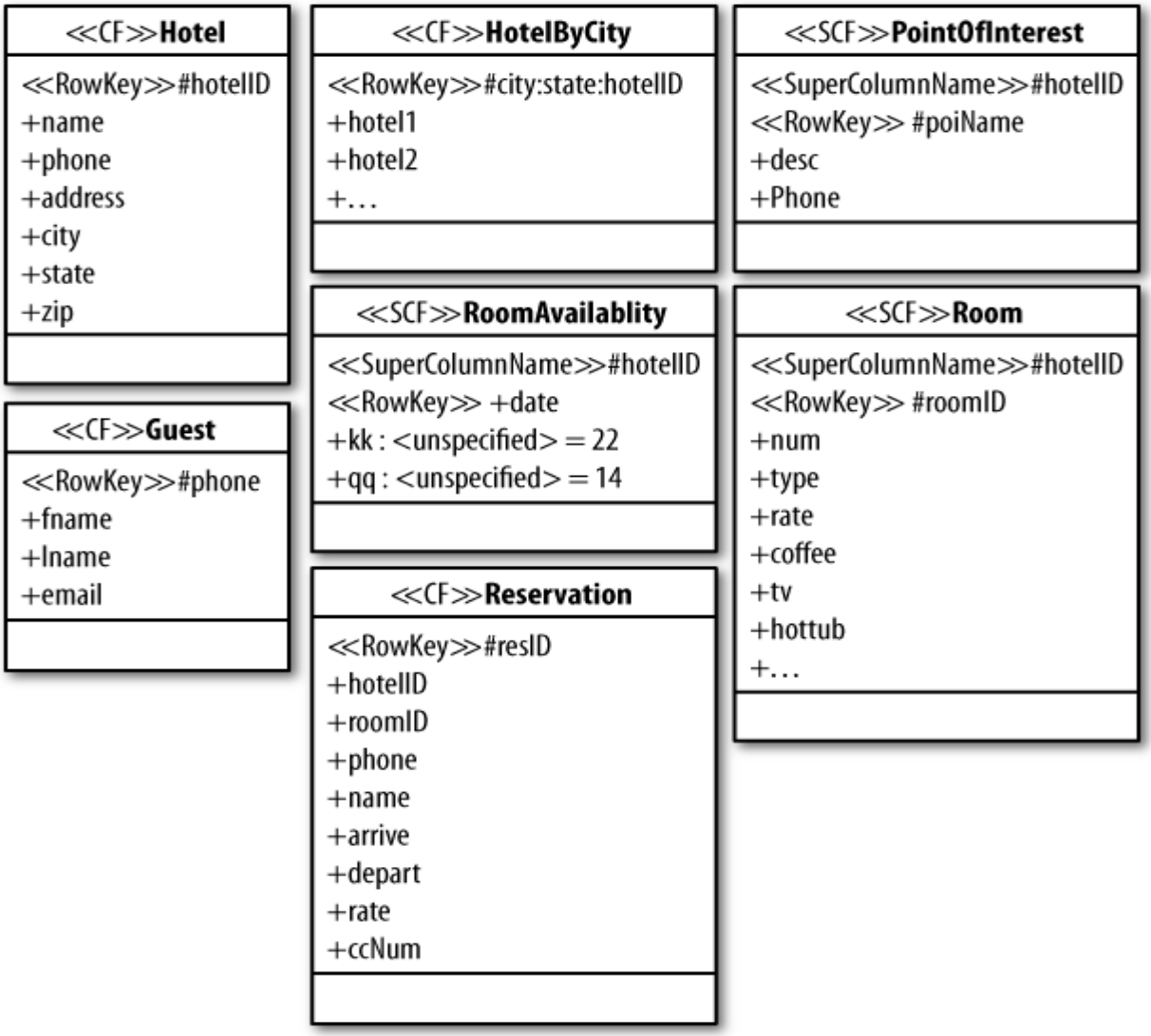

Figure. 2 Hotel Management System

Our application Hotel Management System designed with Cassandra includes the following characteristics:

- Find hotels in a given area.
- Find information about a specific hotel, such as its name, location, room availability etc.
- Find interesting locations near to a given hotel.

- Find availability of rooms in a given date range.
- Find the amenities and rate for a room.
- Possible to book the selected rooms by entering guest information.

The database in Cassandra is created using keyspace. A keyspace in Cassandra is a namespace which defines data replication on nodes. A cluster contains one keyspace per node.The application we're building will do the following things:

1. Create the database structure.
2. Populate the database with hotel and point of interest data. The hotels are stored in standard column families, and the points of interest are in super column families.
3. Search for a list of hotels in a given city. This uses a secondary index.
4. Select one of the hotels returned in the search, and then search for a list of points of interest near the chosen hotel.
5. Booking the hotel by doing an insert into the Reservation column family should be straightforward at this point, and is left to the reader.

6.1. Table Operations

To create a table use the command CREATE TABLE. The tables required for the Hotel Management System application can be created using this command. The syntax is
CREATE (TABLE | COLUMNFAMILY) <tablename> ('<column-definition>', '<column-definition>')
The primary key is represented by a column that is used to uniquely identify a row. Therefore, defining a primary key is mandatory while creating a table. A primary key is also made of one or more columns of a table [4].

6.2. CURD Operations

To create data in a table use the command INSERT. The syntax for creating data in a table is
INSERT INTO <tablename> (<column1 name>, <column2 name>....) VALUES (<value1>, <value2>....)
UPDATE is the command used to update data in a table. The syntax of update is
UPDATE <tablename> SET <column name> = <new value>
<column name> = <value>.... WHERE <condition>
Reading Data using SELECT Clause from a table in Cassandra. Using this clause we can read a whole table, a single column, or a particular cell. The syntax of SELECT is
SELECT FROM <table name> WHERE <condition>
Delete data from a table using the command DELETE. Its syntax is
DELETE FROM <identifier> WHERE <condition>

6.3. Performance Evaluation

One of the hallmarks of Cassandra is its high performance, for both reads and writes operations. When new nodes are added to a cluster, Cassandra scales it linearly. The performance of Hotel Management System application is evaluated with various hardware requirements such as Intel core CPU @ 1.80 GHz, 64-bit operating system, x64 based processor, 4.00GB RAM. The software specifications include Apache Cassandra version 1.2.19. Figure 3 gives performance of Cassandra operations.
In the graph of performance evaluation of Cassandra database X axis represents the throughput in ops/sec and Y axis represents average latency in ms. Here three operations such as update, insert and read are evaluated for performance. In the graph it is clear that update operation has very high

throughput while it is in low latency. Similarly insert operation has high throughput [7] while it is in low latency which is greater than latency of update operation. In the case of read operation which has low throughput while it is in high latency.

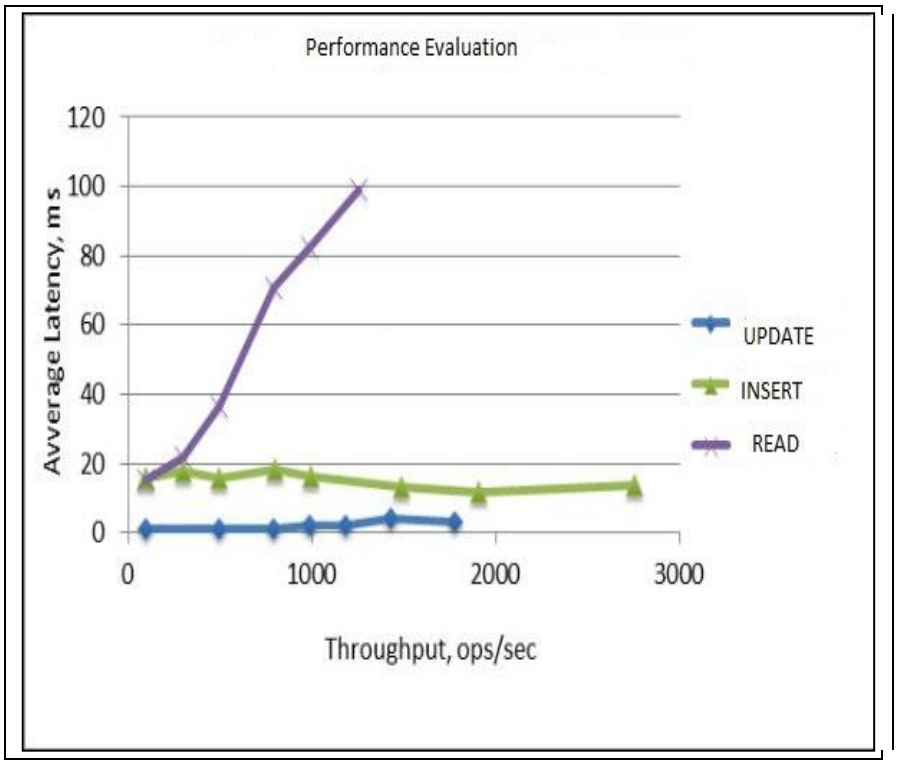

Figure. 3 Performance Evaluation

7. CONCLUSION

NoSQL database: Cassandra is built, implemented, and operated a scalable storage system providing high performance, and wide applicability. Demonstrated that Cassandra can support a very high update throughput while delivering low latency. It is very efficient as compared with other databases.

REFERENCES

[1] http://cassandra.apache.org
[2] http://www.tutorialspoint.com/cassandra/
[3] Dietrich Featherston, Cassandra: Principles and Application, 2010
[4] A. Lakshman, P. Malik, Cassandra - A Decentralized Structured Storage System, Cornell, 2009.
[5] https://www.safaribooksonline.com/library/view/cassandra the definitive/9781449399764/ ch04.html
[6] Robin Hecht Stefan Jablonski, University of Baybreuth " NoSQL Evaluation A Use Case Oriented Survey" 2011 International Conference on Cloud and Service Computing.
[7]Matthias Nicola and Matthias Jarke. Performance modeling of distributed and replicated databases. IEEE Trans. on Knowl. and Data Eng.,12(4):645–672, July 2000.

Authors

Hima S. is currently pursuing M.Tech in Computer Science and Engineering in Mar Athanasius College of Engineering. She completed her B.Tech from Mohandas College of Engineering and Technology, Thiruvananthapuram. Her areas of research are Image Processing, Database and Data Mining.

Varalakshmi P. is currently pursuing M.Tech in Computer Science and Engineering in Mar Athanasius College of Engineering. She completed her B.Tech from P.R.S. College of Engineering and Technology, Thiruvananthapuram. Her areas of research are Data Mining, Databases and Image Processing.

Surekha Mariam Varghese is currently heading the Department of Computer Science and Engineering, M.A. College of Engineering, Kothamangalam, Kerala, India. She received her B-Tech Degree in Computer Science and Engineering in 1990 from College of Engineering, Trivandrum affiliated to Kerala University and M-Tech in Computer and Information Sciences from Cochin University of Science and Technology, Kochi in 1996. She obtained Ph.D in Computer Security from Cochin University of Science and Technology, Kochi in 2009. She has around 25 years of teaching and research experience in various institutions in India. Her research interests include Network Security, Database Management, Data Structures and Algorithms, Operating Systems and Distributed Computing, Machine learning. She has published 17 papers in international journals and international conference proceedings. She has been in the chair for many international conferences and journals.